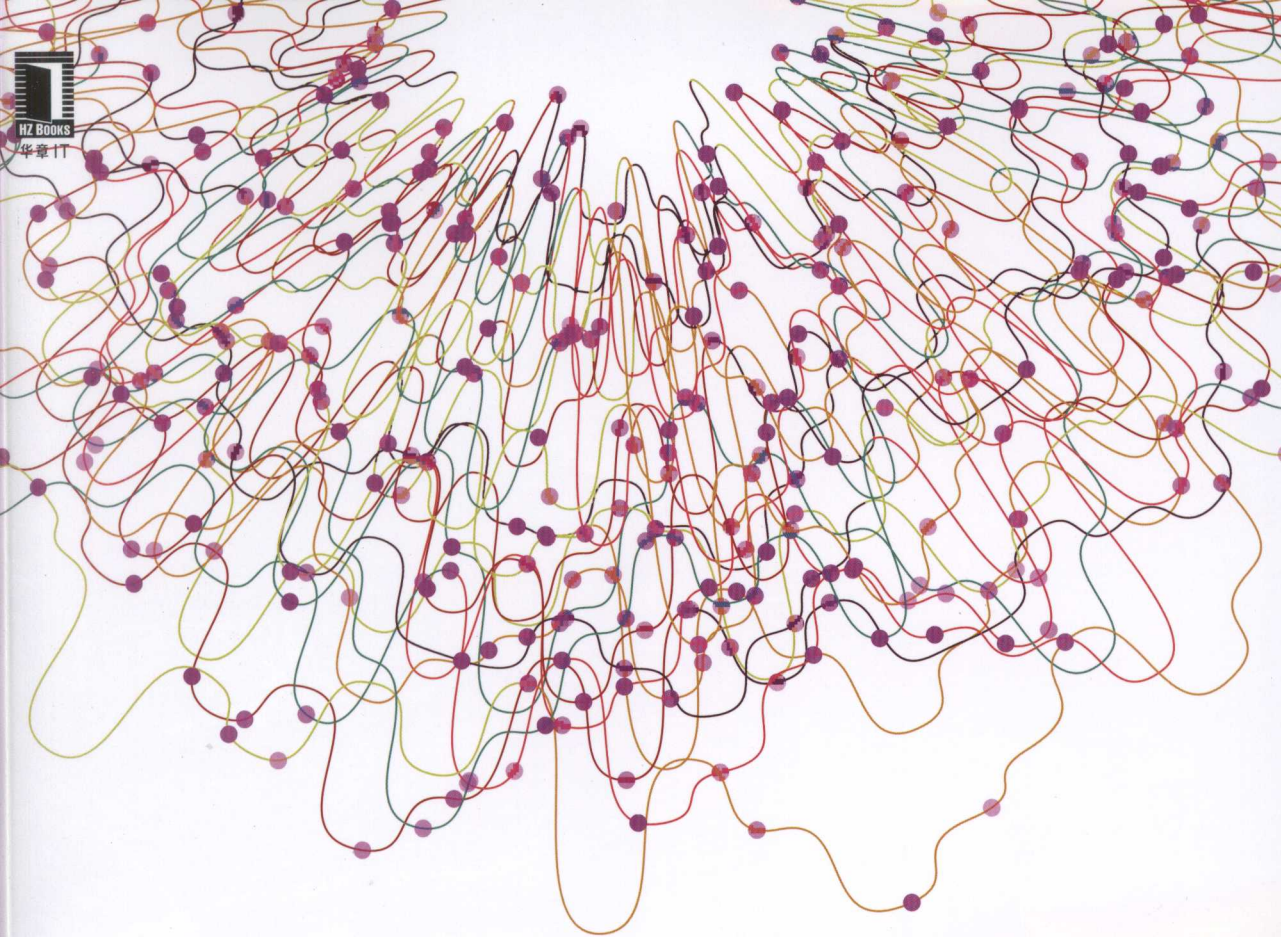


版权注意事项：1、书籍版权归著者和出版社所有；
2、本PDF仅用于个人获取知识，进行私底下知识交流；
3、PDF获得者不得在互联网以任何目的进行传播；
如有需要，请尽量购买正版实体书！支持书籍作者！！



企业大数据系统构建实战

技术、架构、实施与应用

吕兆星 郑传峰 宋天龙 杨晓鹏◎著
尹慧敏◎审校

国内一线大数据专家撰写，多位商界、学术界、技术界大佬联袂推荐！

从战略规划、落地实施、价值提升3个维度，为企业从数据端到应用端
全方位构建大数据系统提供指导，有高度、有逻辑、有实战！



机械工业出版社
China Machine Press

内容简介

对于很多企业而言，大数据的重要性不言而喻，但是如何构建、实施和应用大数据系统却是一个复杂的工程。本书让读者认识到大数据不仅是数据、技术、架构、应用，更是结合了商业模式、战略定位、信息安全、单位协同、组织保障、实施选型的完整体系。

本书内容从大数据的规划定位、组织实施和价值提升三个维度展开，兼顾整体性、全局性、安全性、价值性、技术性、体系性等方面的考虑。

第一部分：企业大数据战略规划

主要从宏观的角度介绍大数据的定位、组织保障、解决方案选择和自主实施思路，目的是从全局角度引导建立大数据工作的整体思维。

第二部分：企业大数据落地实施

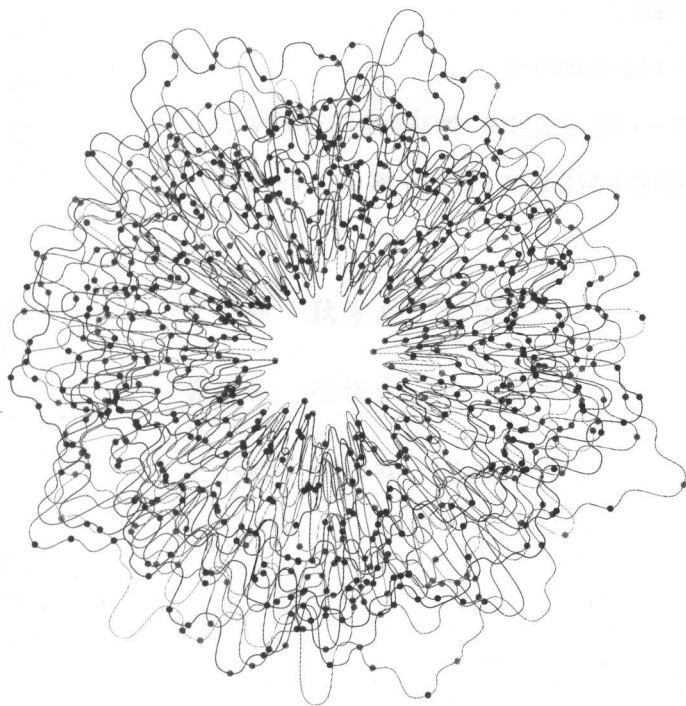
主要从执行层面介绍了与大数据落地相关的技术、架构、开发、工作流、应用和价值评估，直接以落地视角解读大数据工作中每个环节涉及的流程、知识和方法，这部分也是本书的核心。

第三部分：大数据价值、变革和挑战

主要涉及大数据的社会价值、当前问题和挑战，以及大数据的未来趋势，这是对现有大数据工作的延展以及未来趋势的探索。



技术丛书



企业大数据系统构建实战

技术、架构、实施与应用

吕兆星 郑传峰 宋天龙 杨晓鹏◎著

尹慧敏◎审校



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

企业大数据系统构建实战：技术、架构、实施与应用 / 吕兆星等著. —北京：机械工业出版社，2017.5

(大数据技术丛书)

ISBN 978-7-111-56876-6

I. 企… II. 吕… III. 企业管理—数据管理 IV. F272.7

中国版本图书馆 CIP 数据核字 (2017) 第 090179 号

企业大数据系统构建实战：技术、架构、实施与应用

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）

责任编辑：何欣阳

责任校对：殷虹

印刷：北京市荣盛彩色印刷有限公司

版次：2017 年 5 月第 1 版第 1 次印刷

开本：186mm×240mm 1/16

印张：31.5

书号：ISBN 978-7-111-56876-6

定价：89.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010) 88379426 88361066

投稿热线：(010) 88379604

购书热线：(010) 68326294 88379649 68995259

读者信箱：hzit@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问：北京大成律师事务所 韩光 / 邹晓东

特别顾问团队

史 研 狄方春 刘宗旬

黄运豪 陈振宇 王 成

王小鹏 田学峰 姜继浩

Preface 前言

为什么要写这本书

随着 2013 年大数据元年的开启，各行各业都已经将大数据视为推动企业发展、推进行业进步、加快产业升级、促进民生繁荣、巩固社会安全甚至提升国家竞争力的核心武器。从个性化推荐、关联销售到精准营销，从云平台、云服务、云计算到大数据产业链，从百度迁徙、高考预测到冬季流感预测，从机器学习、图像识别到智能交通，从奥巴马总统竞选到美国中央情报局反恐，从美国的大数据研究和发展计划到中国的促进大数据发展行动纲要等一系列事实说明了大数据正受到来自政治、经济、社会、文化、军事等各个领域的广泛关注，并越来越彰显其巨大价值。

大数据不仅是一个技术名词，更是当下企业资产、核心竞争力、完整产业链和先进生产力的代名词。因此，大数据应该是作为一个整合概念和体系被认知，而非独立的方法论、技术论甚至应用论。处于飞速变革时代的中国，在大数据产业链各个环节的企事业单位受限于自身产业属性、盈利模式、利益趋向、认知、能力等，无法完整地展示出大数据的知识图谱与价值图谱。

纵观当下整个大数据认知取向，大致有三类基本认知点：

第一类是大数据知识论，这种认知以大数据方法、理论、知识的研究和推导为聚焦点，通过深度学习，归纳、总结出大数据知识体系。这是典型的学院派，优势是对基础理论研究非常透彻并且具备深厚的理论基础，不足之处是缺乏对产业、学术、应用的结合，更缺少真正能落地的应用案例。

第二类是大数据技术论，这种认知以大数据技术为聚焦点，落脚于大数据的硬件、服务、架构、开发、计算、算法等具体实施层面。诚然，大数据技术是大数据实施的核心，也是带来技术变革和生产力突破的关键，但只有技术而缺乏正确的方向以及有价值的应用引

导，技术便无法发挥作用，更无法转化为经济价值、社会价值和政治价值。

第三类是大数据应用论，这种认知以大数据的场景化为聚焦点，通过对历史、现在、未来的变革、创新和实践的总结和构想，营造出大数据的丰富应用场景和能力空间。这是一种典型的以应用为驱动的认知理论，通过落地案例驱动技术来表现大数据的巨大价值。但这种应用论过于专注场景化包装，更强调落地而忽视技术的巨大潜力和推动作用，更无法体现出大数据作为企业资产、技术竞争力等非直接利润表现的价值因素。

本书的几位联合作者彼此是共事多年的朋友，各自负责大数据工作中的不同环节。大家的工作和知识有交集更有互补，因此，我们认为只有依靠这种“知识合并”和“知识互补”的关系才能够呈现出大数据的全貌，这也是撰写本书的出发点之一。

当前，市场上有非常多关于大数据的书籍，但能从整体性、全局性、安全性、价值性、技术性、体系性等方面完整考虑的书非常少。我们希望通过本书让读者认识到大数据不仅仅是数据、技术、架构、应用，更是结合了商业模式、战略定位、信息安全、单位协同、组织保障、实施选型的完整体系。

几位联合作者对于本书内容的贡献如下：吕兆星撰写了技术的架构部分，包括第4章、第5章、第6章；郑传峰撰写了战略和应用的部分，包括第1章、第2章、第9章；宋天龙撰写了数据和价值评估的部分，包括第3章、第8章、第10章、第11章、第12章、第13章；杨晓鹏撰写了技术开发的部分中第7章的全部内容。

作者简介

吕兆星 (Ethan Lv)

资深大数据技术专家，精通基于大数据的分布式数据挖掘、存储、计算技术，以及其生态体系架构；精通垂直搜索技术、机器学习、文本情感倾向性挖掘、网络爬虫、全文索引体系架构。曾任软通动力集团大数据研究院总架构师、HiveCloud 创始人、萝卜网 CTO、国美在线大数据中心高级架构师等。

主导研发的大数据和文本挖掘平台包括：DMP、DSP、推荐系统、决策运营系统、iCreations 系列产品、蜂梭系列产品、军犬舆情系列产品等。并成功应用到能源、电力、电商、电信、金融、政府、食品、医疗保健等行业与机构，覆盖超过 500 家各级企业用户。所著《基于机器学习的数据挖掘模型》获得国家级技术创新基金。

郑传峰 (Peter Zheng)

大数据业务应用领域专家，主导大数据方向的战略规划，包括数据产品、数据应用、数据价值变现等方向。曾任软通动力数据科技公司资深数据应用专家，HiveCloud 首席战略官。

阶段性负责国美电器、国美在线，以及库巴网会员营销、网站运营和产品设计工作，在 CRM 系统、DMP 数据平台、精准营销系统、广告精投、能源大数据拥有多年的操盘经验，涉及大数据上层应用服务产品的设计、咨询和实施。参与多家大型企业或机构的大数据战略规划和实施，行业覆盖零售、电商、电信、政府、交通、能源和电力等。

宋天龙 (Tony Song)

大数据领域资深数据分析、挖掘和建模专家，精通端到端数据价值场景设计、业务需求转换、数据结构梳理、数据建模与学习，以及数据工程交付。曾任软通动力集团大数据研究院数据总监，Webtrekk（德国最大的网站数据分析服务提供商）中国区技术和咨询负责人，国美大数据中心经理。

拥有丰富的项目工作经验，参与过集团和企业级大数据存储平台、大数据开发和集成平台、数据体系规划、大数据产品开发、网站流量系统建设、网站智能推荐、企业大数据智能等大型数据工作项目。参与实施客户案例包括 Webpower、德国 OTTO 集团电子商务（中国）、Esprit 中国、猪八戒网、顺丰优选、乐视商城、泰康人寿、酒仙网，国美在线、迪信通等。合作培训及沙龙单位包括人民大学、数盟、萝卜网、Netconcepts、触脉、中商联数据分析委等。萝卜课堂、天善学院特邀讲师，百度文库认证作家，36 大数据、站长之家、互联网分析沙龙专栏作家。著有《网站数据挖掘与分析：系统方法与商业实践》一书。

杨晓鹏 (Kelvin Yang)

大数据及 BI 技术领域资深架构师，精通传统数据模式及大数据分布模式的数据存储、计算与应用架构，以及大数据量的数据迁移、存储、索引、计算、分析与挖掘等相关环节的设计、实现与优化。曾任软通动力集团大数据研究院高级架构师，HiveCloud 总架构师，主导大数据存储平台、计算平台和应用服务平台的设计与研发，曾任居然之家 O2O 大数据平台总负责人、中国银联大数据报文分析项目高级技术顾问、国美在线大数据中心高级技术工程师。

曾参与企业级项目包括大型电商网站的 BI 系统、数据仓库、大数据系统等设计和研发项目，以及金融银行类企业风险及异常交易分析项目。实施大中型企业数据项目包括居然之家、中国银联、华农保险、中国电信等超过 50 家客户的案例。精通大数据 Hadoop、Hive、HBase、Impala、Spark 等组件的架构与实施，精通数学模型，自主开发实现分治 / 覆盖的 C4.5 决策树、马尔科夫预测、KMeans、Apriori 等模型算法程序，并成功应用到电商、金融等行业。

读者对象

本书虽然是一本有关大数据的书籍，但并没有对读者的数据、技术等专业知识做硬性要

求,相反,我们尽量让书籍的内容深入浅出、便于理解。当然,如果读者具有一定的知识背景,在对专业知识的理解上会更有帮助。本书适合以下几类读者阅读:

- **对大数据感兴趣的专业人员。**数据工作能力已经成为提升自身技能、增强职业竞争力的重要因素。无论读者从事什么工作,如果能够将大数据的思路、价值和应用方法与工作实践相结合,一定会对现有工作有所帮助。
- **刚进入大数据行业的新人。**刚入行的行业新人需要对大数据有完整的认知,然后才能针对不同的大数据工作并结合自己特点、喜好等制定适合自己的职业规划和成长路径。本书针对大数据体系做出详细、系统的介绍,涵盖从战略规划到实施应用,从技术架构到技术开发,从数据工作流到价值评估等一系列知识,对新人的指导意义非常大。
- **具备一定实践经验的大数据从业者。**对于已经在大数据方面工作1~3年的从业者,相信你已经遇到了一些瓶颈,想要在原有的大数据思维基础上获得更有效的工作方法和价值提升。本书中丰富的应用案例可以帮助你拨开云雾见青天。
- **已经具有丰富工作经验的大数据从业者。**当大数据从业者工作3年以上时,就已经有机会从执行层走向管理层。机会总是留给准备好的人,作为管理者如何从数据工作流程、制度、风险、绩效、安全和价值等方面进行思考并开展工作?相信本书会给你满意的答案。

如何阅读本书

本书内容共分为三个部分,按照大数据的规划定位、组织实施和价值提升,以及变革与挑战的思路撰写。

第1~4章讲解企业大数据的战略规划,主要从宏观的角度介绍大数据的定位、组织保障、解决方案选择和自主实施思路,目的是从全局角度引导建立大数据工作的整体思维。

第5~10章讲解企业大数据的落地实施,主要从执行层面介绍了大数据落地的相关技术、架构、开发、大数据工作流、应用和价值评估,直接以落地视角解读大数据工作中每个环节涉及的流程、知识和方法,这也是本书的核心章节。

第11~13章讲解大数据的价值、变革和挑战,主要涉及大数据的社会价值、当前问题和挑战以及大数据的未来趋势,这是对现有大数据工作的延展以及未来趋势的探索。

由于本书各个章节的内容相对独立,均可自成体系,因此在阅读本书的过程中并不要求读者注意特定的逻辑关系,读者可直接选择感兴趣的内容阅读。但是,从整书的逻辑结构和撰写出发点上,仍然建议读者从头开始阅读。

勘误和支持

由于作者的水平有限，书中难免会出现一些错误或者不准确的地方，恳请读者不吝指正。为此，作者特地创建了一个QQ群（群号：303237546），读者可以在QQ群中进行交流并提出意见与建议（或者添加微信 TonySong2013 进行反馈）；同时如果有任何问题，也可以在群中沟通讨论；更重要的是，我们希望能够将从事大数据行业的志同道合的人士聚集起来，分享彼此的工作经验。

致谢

在本书的撰写过程中，我们得到了来自多方的指导、帮助和支持。

首先要感谢的是我们的创作导师及本书的审校尹慧敏先生，以及软通动力集团高级副总裁史研先生，是他们给我们提供了更多探索企业大数据在不同行业的实践机会，并在该过程中给予我们战略引导、思想提升和方法启迪，本书的顺利完成与此密不可分。

其次要感谢在各个大数据项目 and 工作中，担当核心骨干的团队成员，他们是王平、曹佳佳、陈骏、陈海洋、李国彬、吕奔、姚璐、张丽涛、江涛、豆阿婷、高杰、侯良伟、杨勇、麻建昕等；当然，也要感谢来自项目团队中的各位领导、伙伴、朋友的大力支持！

再次要感谢机械工业出版社华章公司的杨福川老师，是他鼓励我们完成了本书并在撰写过程中给予了详细的思路拓展和专业指导；感谢全程参与审核、校验等工作的孙海亮编辑以及其他在背后默默支持的出版工作者。

最后要感谢我们的家人和朋友，在写书的这段期间里，他们帮我们解决了很多生活和工作中的问题，使得我们有精力、有时间完成本书的全部撰写工作。

谨以本书献给我们最亲爱的父母，以及众多热爱大数据工作并为之奋斗的朋友们！

宋天龙 (Tony Song)

Contents 目 录

前言

第1章 企业大数据战略定位 1

1.1 宏观 1

1.2 微观 4

1.2.1 资源协同 5

1.2.2 战略定位 6

1.2.3 启动契机 7

1.2.4 大数据历程 9

1.3 本章小结 12

第2章 企业大数据职能规划 13

2.1 大数据组织架构体系 13

2.1.1 大数据部门在企业中的角色 13

2.1.2 常见的大数据职能及职责 17

2.2 大数据职位构建体系 24

2.2.1 基础平台类 24

2.2.2 数据管理类 26

2.2.3 技术研发类 27

2.2.4 产品设计类 30

2.2.5 数据挖掘类 32

2.2.6 数据分析类 33

2.3 大数据制度和流程规范 35

2.3.1 制度和流程规范意义 35

2.3.2 制度和流程规范内容 35

2.3.3 制度和流程规范模板 42

2.4 本章小结 44

第3章 企业大数据解决方案 45

3.1 企业大数据解决方案实现方式 45

3.1.1 独立研发 45

3.1.2 第三方解决方案 46

3.1.3 联合开发 57

3.2 如何选择解决方案 58

3.2.1 外部环境分析 58

3.2.2 内部环境分析 59

3.2.3 需求规划分析 62

3.2.4 解决方案特性分析 63

3.2.5 解决方案费用评估 67

3.3 本章小结 70

第4章 企业大数据自主实施思路 71

4.1 制定规划原则 71

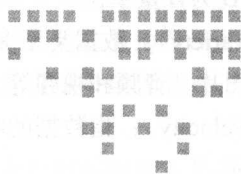
4.1.1 价值性 71

4.1.2 实时性	72	5.1.1 Hadoop 生态	115
4.1.3 高效性	72	5.1.2 NoSQL	142
4.1.4 安全性	72	5.1.3 实时计算	150
4.1.5 延展性	73	5.1.4 全文检索	160
4.1.6 全局性	74	5.2 相关技术	204
4.2 制定目标蓝图	75	5.2.1 数据可视化	204
4.3 制定建设目标	76	5.2.2 数据缓存	220
4.4 明确组织规划	78	5.2.3 中间件	227
4.4.1 组织结构设计的作用	79	5.2.4 关系型数据库	236
4.4.2 组织结构设立的导向	79	5.2.5 数据 ETL	245
4.4.3 组织结构的最终设立	81	5.3 大数据算法库	250
4.5 设计技术方案	85	5.4 本章小结	276
4.5.1 大数据系统建设方案	85		
4.5.2 大数据系统与传统 BI 的 融合方案	91	第 6 章 大数据架构设计	277
4.6 制定人才规划	94	6.1 大数据架构设计原则	277
4.6.1 指导思想	94	6.2 大数据核心架构要素	279
4.6.2 规划原则	94	6.3 大数据架构设计模式	284
4.6.3 核心内容	95	6.4 本章小结	289
4.7 投入产出评估	97		
4.7.1 数据投入与产出的内涵	97	第 7 章 大数据技术开发	290
4.7.2 数据投入与产出的特征	98	7.1 数据采集	290
4.7.3 数据投入与产出的管理	99	7.1.1 批量采集	291
4.8 数据风险管理	105	7.1.2 增量采集	292
4.8.1 数据风险管理的概念	105	7.2 数据存储	293
4.8.2 数据风险管理的类型	106	7.2.1 HDFS 文件存储引擎	294
4.8.3 数据风险管理的原则	109	7.2.2 Hive 数据存储引擎	295
4.8.4 数据风险管理与控制	110	7.2.3 HBase 列式存储引擎	295
4.9 本章小结	114	7.2.4 MySQL 关系型数据存储引擎	296
第 5 章 大数据技术介绍	115	7.3 多维计算	296
5.1 核心技术	115	7.4 功能服务	299
		7.5 平台管理	301

7.5.1 监控管理	301
7.5.2 调度管理	302
7.5.3 权限管理	304
7.6 应用域	307
7.7 本章小结	308
第 8 章 大数据 workflow	309
8.1 数据源	310
8.1.1 日志 / 文件	310
8.1.2 数据库	310
8.1.3 网络爬虫	311
8.1.4 第三方 API / 合作	311
8.2 数据处理	312
8.2.1 数据质量校验	312
8.2.2 清洗转换	316
8.2.3 质量提升	320
8.2.4 数据脱敏	321
8.2.5 集成整合	323
8.3 数据存储	324
8.3.1 关系型数据库	324
8.3.2 分布式文件系统	325
8.4 数据计算	325
8.4.1 三种数据计算时效性	325
8.4.2 结构化数据计算	327
8.4.3 半 / 非结构化数据计算	333
8.4.4 深度挖掘学习	360
8.5 数据应用	376
8.5.1 辅助决策	376
8.5.2 数据驱动	377
8.6 数据质量管理	379
8.6.1 数据质量建设的内涵	379
8.6.2 影响数据质量的常见因素	380

8.6.3 数据质量建设的框架	381
8.7 本章小结	392
第 9 章 企业大数据业务应用	393
9.1 大数据应用场景概述	393
9.1.1 场景商业目的分析	394
9.1.2 场景数据来源分析	394
9.1.3 场景数据难易分析	397
9.1.4 场景应用举例	397
9.2 用户画像	407
9.2.1 业务应用背景	407
9.2.2 主要实现过程	408
9.2.3 关键应用场景	414
9.2.4 应用价值提炼	415
9.2.5 场景总结回顾	417
9.3 个性化营销	419
9.3.1 业务应用背景	419
9.3.2 主要实现过程	421
9.3.3 关键应用场景	424
9.3.4 应用价值提炼	425
9.3.5 场景总结回顾	426
9.4 精准广告	427
9.4.1 业务应用背景	427
9.4.2 主要实现过程	429
9.4.3 关键应用场景	438
9.4.4 应用价值提炼	439
9.4.5 场景总结回顾	440
9.5 征信	441
9.5.1 应用场景背景	441
9.5.2 主要实现过程	442
9.5.3 主要应用场景	447
9.5.4 应用价值提炼	449

9.5.5 场景总结回顾	449	11.2 政务价值	465
9.6 本章小结	450	11.3 产业价值	468
第 10 章 企业大数据价值评估	451	11.4 本章小结	470
10.1 资产价值	451	第 12 章 大数据当前问题及挑战	471
10.1.1 数据规模	451	12.1 数据挑战	471
10.1.2 数据价值度	452	12.2 安全挑战	472
10.1.3 数据鲜活性	454	12.3 价值挑战	474
10.1.4 数据关联维度	454	12.4 认知挑战	475
10.1.5 数据粒度	455	12.5 技术挑战	478
10.2 业务价值	455	12.6 人才挑战	480
10.2.1 用户体验提升	455	12.7 本章小结	481
10.2.2 运营优化	457	第 13 章 大数据未来趋势	482
10.2.3 销售贡献	460	13.1 价值资产化	482
10.2.4 供应链优化	461	13.2 产业生态化	487
10.3 本章小结	462	13.3 主体社会化	490
第 11 章 大数据的社会价值	463	13.4 应用智能化	491
11.1 民生价值	463	13.5 本章小结	492



企业大数据战略定位

企业大数据的战略定位，决定了企业大数据发展的可行性、持续性、稳定性和高效性，但如果要明确大数据战略定位，我们首先要了解什么是大数据，大数据平台技术与传统数据库的区别是什么？为什么要做大数据，大数据可以解决什么问题？

本章将从宏观和微观两个层面介绍企业大数据的战略定位，试图剖析如何将大数据摆在企业发展的正确位置上，以及如何统筹不同资源协同大数据的工作关系并最大化大数据价值。

1.1 宏观

大数据定义多种多样，其中较为典型的有：

研究机构 Gartner 给出了这样的定义：需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力来适应海量、高增长率和多样化的信息资产。

麦肯锡全球研究所给出的定义：一种规模大到在获取、存储、管理、分析方面大大超出了传统数据库软件工具能力范围的数据集合，具有海量的数据规模、快速的数据流转、多样的数据类型和价值密度低四大特征。

笔者认为：大数据是指无法用常规数据工具软件进行获取、存储、计算和管理的数据集合，是需要新 IT 技术才能使其具备更好的洞察发现力、流程优化能力，并提供智能决策力，以此来适应大量、高增长、多样化和有价值的信息资产。

大数据具有如下特征：

□ 容量 (Volume)：传统数据库容量一般以 MB 和 GB 为计量单位，而大数据是以 GB、

TB 和 PB 为计量单位。

- ❑ 种类（Variety）：数据类型多种多样，包含结构化、半结构化和非结构化，例如文本、日志、图片、音频和视频等。
- ❑ 速度（Velocity）：指数据的响应速度，包括数据获取速度和数据输出速度都要优于传统数据库。
- ❑ 真实性（Veracity）：大数据存在噪声较多，需要经过筛选、填充和删除的过程，确保数据的真实性和有效性。
- ❑ 复杂性（Complexity）：数据量巨大，来源多渠道，包括自有数据、网络数据、合作伙伴数据，同时数据质量和类型又加深了大数据的复杂性。
- ❑ 价值（Value）：大数据蕴藏着既定的价值，价值程度取决于应用数据的对象，通过合理的商业目标即可挖掘出数据潜藏的金矿。

大数据平台技术与传统数据库的差异如表 1-1 所示。

表 1-1 大数据平台技术与传统数据库的差异

功能项目	传统数据库	大数据平台
数据规模	小（以 MB 单位为主）	大（以 GB、TB 和 PB 单位为主）
数据类型	单一数据结构类型	可连接多个数据来源，多种类型，包含 Oracle、MySQL、文本、日志等主流数据库的结构化、半结构化和非结构化的数据问题。非结构化数据包括文本、音频、视频和图片等
数据范围	部分样本（部分库表或业务单元的联系）	全量数据（全部库表 and 所有业务内容建立联系）
数据算法	加减乘除为主，计算效率低	分类、聚类、回归、神经网络和时序等各种算法
数据处理	一刀切的方式	没有定式，离线计算和实时计算，根据数据源类型选择更适合的方式
数据应用	统计和对比为主	不限时间、空间各维度的分析与对比，可为多种不同的应用灵活地提供数据调用与展示

大数据具有很多传统数据库不具备的优点，那么大数据可以解决哪些问题？

- ❑ 连接数据孤岛：将企业各个孤立的信息孤岛进行连接，实现数据信息正向和反向的查询，由原来的单一信息查阅，变为全景式的鸟瞰企业数据内容。
- ❑ 整合信息资源：通过虚拟化技术，整合 IT 信息资源，有效地展现软硬件和网络资源的使用和计算情况，更加合理地规划和使用 IT 资源。
- ❑ 内部效率提升：通过信息孤岛的连接，缩短了往常数据提取、存储、整合和计算的时间，根据业务需求的难易程度，通过合理分配离线和实时计算，能够大幅度提高内部效率。
- ❑ 供应链优化：数据连接不是局限于企业内部，而是延伸到企业的上下游，为合作伙伴提供数据共享平台，有利于提升供应链上下游的协同合作，进一步提升供应链效率和效果。

- ❑ 企业业绩增长：大数据的模型算法包含有指导和无指导两种类型，有指导的算法能够帮助企业优化现有的业务流程，从中找到业务规律，更有效地帮助业绩增长，而无指导的算法结合全景式的数据，可以帮助企业找到业务增长或创新点，更好地帮助企业拓宽业务范畴。
 - ❑ 用户体验提升：用户包含内部和外部，内部指企业自有的管理人员以及员工，而外部则包含企业面对的客户，以及供应链上下游的合作伙伴，通过大数据全景式的数据集市，为企业用户提供更加完善和流畅的服务，有助于提升整体的体验。
 - ❑ 产业服务全景化：企业除了纵向地与上下游伙伴进行数据共享，还有另一种状态，即通过横向与其他伙伴进行数据连接，从而实现全产业链的全景数据化。因为大数据最重要的理念是开放、共享和协作，只有连接更多有效有价值的数据才能使企业甚至整个产业屹立不倒。
- 了解了大数据能够解决的问题，那么企业的大数据战略应该是什么样子的，应该如何对企业大数据战略进行定位？我们需要考虑以下问题，如图 1-1 所示。

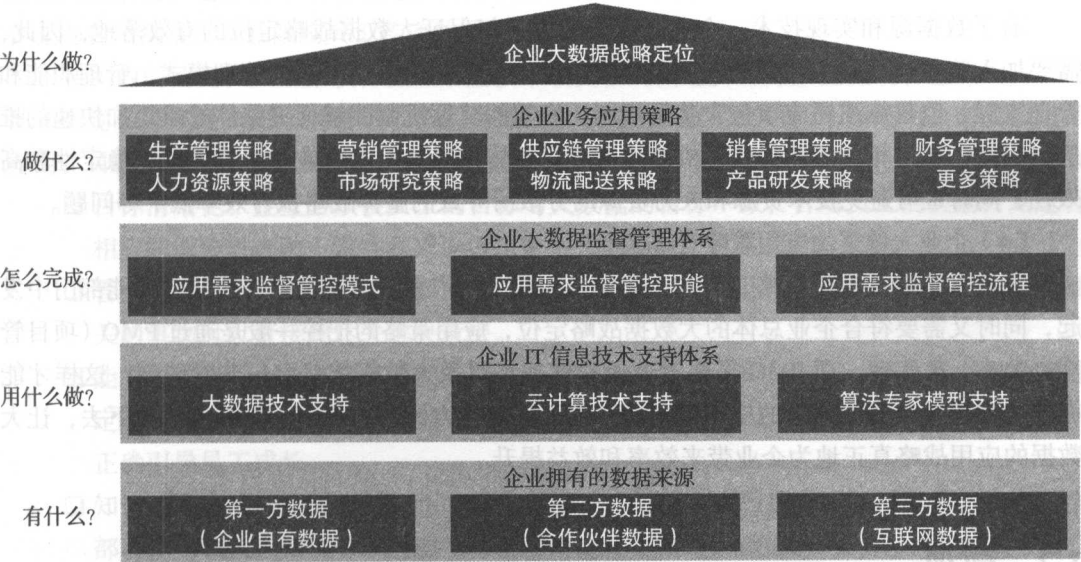


图 1-1 企业大数据战略定位

企业大数据战略定位可以自下而上或者自上而下地进行确定，两种方式各有利弊，自下而上容易造成总体战略不明确，从而导致整体战略定位的失败，但自下而上的方式通常是更容易落地执行的办法；自上而下则相反，根据企业在市场中所处的地位、竞争对手情况、宏观经济环境、供应链状态和市场用户的实际需求，大数据的总体目标更能符合企业总体发展战略，但如果战略没能贯彻到底或者在没有获得认可的情况下，这个战略最后是很难落地执行的。因此通常是以自上而下为主，通过自下而上的办法来修正总体大数据战略定位的方

式。不论企业最终以哪种方式来确定大数据的战略定位，以下内容都是必须在制定战略时考虑的：

（1）考虑企业有什么数据

数据分为三种类型，第一方数据（企业自有数据）、第二方数据（合作伙伴数据）和第三方数据（互联网数据），企业需要评估自身具备的数据条件（包括数据来源渠道可靠性、数据更新及时性、数据质量有效性、数据内容完整性等），来考虑或者设计规划总体大数据的战略定位，正所谓巧妇难为无米之炊，数据是大数据战略的根本，不清楚数据源的情况，就无法确定整体的大数据战略定位。

（2）通过哪些技术实现

大数据的处理技术多种多样，有硬件虚拟化技术、数据存储技术、数据检索技术、数据计算技术、数据挖掘技术和分布式协调技术等，同时每种技术中也包括各种不同功能的组件，企业应该根据自身的实际需求来选择，盲目贪多，容易导致架构不清晰，甚至影响整体的运行效率，拖累企业大数据战略进程。

（3）如何保证大数据顺利完成

有了数据源和实现技术，企业还需要考虑如何保证大数据战略定位的有效落地，因此，需要加入监督体系，主要是从应用需求的角度出发，明确应用需求的管理模式、管理职能和管理流程，监督体系还应该包含奖惩机制，有效的奖惩机制能够促进整体战略更加快速的推进。因此，通过相应的监督管理机制才能确保整体战略执行的有效性、持续性、稳定性和高效性，同时也可避免技术资源和人力资源的分散所导致的整体战略执行效率低下等问题。

（4）企业大数据的应用策略有哪些

企业大数据的应用策略是与各个职能部门相关的内容，应用策略需要从各职能部门中发起，同时又需要符合企业总体的大数据战略定位，应用策略的把控一般是通过 PMO（项目管理办公室）来实现，而 PMO 需要对企业总体战略以及大数据战略定位非常清晰，这样才能有效地把控各个职能部门的应用策略能够在遵循总体方针的前提下，有效地执行下去，让大数据的应用战略真正地为企业带来效率和效益提升。

1.2 微观

企业大数据项目不可能仅仅依靠一个人或者几个人就可以完成，而是需要一个相互配合的团队，同时要结合各业务部门、IT 部门的不同长处，考虑内部需求和外部环境来综合协同作战，这样才有可能成功。

如何构建大数据团队并推进其落实和部署呢？

要让大数据战略在企业范围内得到最有效的落实和部署，很明显需要一个强有力的大数据团队，进行明确目标、共同协作、协调和信息的分享。然而不明显的是，往往企业会存在一定的阻力，即对于建立大数据组织构建、确立同一目标和沟通协作所需要的变革。

尤其是当要进行深层次的行动,例如大数据需要对组织结构和绩效考核进行优化或者再造时,就不能低估一个企业内抵制变革的能力和意志。国内企业更是如此,典型的有家族式的民营企业,大型的国有企业,甚至涵盖一些外资企业。由于中国所特有的关系和帮派而形成了大量的对变革恐惧进而抵制的人群,因为这可能已经影响到他们的利益或者位置。

1.2.1 资源协同

在大数据工作中提及内部资源,我们经常会想到数据资源、服务器资源、人力资源、费用预算,其实总结起来就是人、财和物,所谓的人即人力资源,我们可以调配、使用或影响到员工群体;财即开展大数据我们拟投入的费用预算,费用预算包含软件成本、硬件成本和人工成本;而物即指我们使用的工具和要加工的对象。

在内部资源的协同上,不同企业会遇到不同的问题,有的是数据问题,有的是服务器问题,有的是费用预算问题,但归根结底更多的是人的问题,人在内部资源的协同上具有至关重要的核心地位。企业员工的行为直接构成或破坏了企业大数据的战略,大数据行动必须不断地关注数据,也需要不断地通过数据指引业务发展、技术开发等各方面细节,令数据深入企业的各个角落,并且激励员工传递数据价值。

在国内的情况,大数据项目往往不是技术或流程的问题,对许多企业或者组织而言,行动和态度必须改变,并且应当促进在企业的协作中对大数据的支持。下面是四个需要改进的方面:目标清晰化、管理扁平化、能力可量化、知识透明化。

❑ 目标清晰化:企业必须使员工的目标与大数据的目标一致,针对员工的奖励计划进行相应的调整对这种一致性和减少组织中主要的抵制力量是非常重要的。

❑ 管理扁平化:战略管理层需要建立一个更加宽松、扁平化的组织,尤其是与大数据相关的事物可以直接到达关键决策层。

❑ 能力可量化:企业必须发展相关的业务技能如大数据分析、挖掘、团队建设和数据评定等,并尽可能地推动内部资格评定与激励,令每位员工都了解自身的优势和劣势,正确引导员工成长。

❑ 知识透明化:应用成熟的知识管理工具和技术以减少大数据分享中的障碍,构建起内部协作与创新,知识分享透明化将会帮助企业大数据快速推进。

在内部资源协同是面临很大挑战的,关于之前企业内部对于大数据的抵制阻力的警告并非危言耸听,那么其后面隐藏着什么呢?永远不要低估组织内对变革的抵制力量,那可能会摧毁团队的一切努力,并导致整体项目走向倒退。

以下是根据历史经验归纳总结的,大数据项目不能轻易实现的原因:

❑ 组织的共同目标不明确;

❑ 组织不鼓励和奖赏好的个人服务;

❑ 没有建立相关激励和知识的数据;

❑ 变革加快,打破了更多人的安逸生活;

- ❑ 大数据被认为只与分析或职能人员相关；
- ❑ 在战略和信息共享上与合作伙伴的协作太少；
- ❑ 缺乏良好的技能，相关的专业资格得不到认可；
- ❑ 企业成员对于大数据定义的理解和实践不在同一水平上；
- ❑ 创造力被“我们不这样做”或者“以前我们不是这样做”的态度扼杀；
- ❑ 在大数据的变革过程中，新规则威胁了旧的权利基础和势力范围；
- ❑ 大多企业的旧传统是重视个人奖金，因此知识分享不能受到激励；
- ❑ 决策者被短期思想驱动，在短时间内看不到效果便被反对者重新引导回去。

综上所述，其实内部资源的协调还是落在了人的问题，只要企业中人的思想观念问题得到解决，资源协调问题就解决了一大半，但是往往在观念的培养上需要一定时间和周期。因此在企业快速实施和上线的过程中，明确的大数据目标、自上而下的扁平管理、可量化的评估体系以及透明的知识管理，可以在一定程度上帮助企业解决观念的问题，以此快速地将企业内外部资源流转起来，共同实现大数据的最终目标。

1.2.2 战略定位

大数据功能如何定位呢？是服务于企业员工、管理层、供应商、客户，还是服务于其他机构？大数据具体的功能地位取决于企业大数据的目标，以及期望通过大数据帮助企业实现什么。从外部服务对象的角度看，大数据功能主要是服务于互联网、企业、政府和民生；然而从内部服务对象的角度看，大数据功能主要是服务于客户、企业自己、合作伙伴，甚至企业所在的行业及行业上下游对应的产业链。因此在实施大数据前，企业要考虑大数据对应的短期和长期战略目标，通过战略目标的确定，定位大数据要帮助企业实现哪些需求场景，这样才能快速有效地助力企业快速发展。

企业级大数据根据企业所处的不同发展阶段，有针对性地制定不同的功能定位，按照功能定位，将技术开发资源进行有策略的倾斜，逐步实现企业级大数据的功能场景。

从大数据工作的角度出发，大数据的功能定位则包含打通数据孤岛、理解数据结构、建立数据标准、建设数据模型以及模型投入商用五个部分。

- ❑ **打通数据孤岛：**依据 Hadoop 分布式大数据平台开发的数据同步功能，能够将结构化（规范的数值表格）和非结构化（文本、日志、音频和图片等）进行一站式的抽取、整合与存储。
- ❑ **理解数据结构：**数据对于每个大中型公司都存在库多表多和逻辑多的情况，尤其对于电商零售公司，很难找到对所有库表的关系非常清楚的一个人，因此需要通过数据血缘关系功能建立数据库之间的关系、数据表之间的关系和数值本身的关系，贯穿业务规则，这样才能最高效地提升运营效率。
- ❑ **建立数据标准：**数据的传输、记录和处理总是会因为各种主观或者客观原因导致不准确或者缺失的情况，需要通过数据的过滤、填充、合并、筛选、分类汇总、排序和计

算等处理功能，建立干净数据，这样才能更加准确地反映企业的真实情况，同时在后续模型建设中才能更好地提炼企业数据价值。

- **建设数据模型：**数据模型是数据挖掘最具价值和魅力的功能，需要融合分类、聚类、回归、时序、神经网络和关联等算法，根据实际业务目标，选择适合的算法进行模型建设，从而最真实地综合和预测业务发展规律。
- **模型投入商用：**模型的最终目标都是应用，对于企业就是商用，实现商业价值，模型的商用目前在营销、精准广告、社交扩展、地理服务、市场研究、金融征信、风险管控、人力资源、信息协同和财务领域的应用相对成熟，企业需要根据自主的业务类型，选择和定位大数据功能，最重要的是能够帮助企业和客户实现对应的商业价值（可以是销售业绩提升，也可以是成本的下降或者利润率的提高等）。

针对电商和零售企业，站在企业经营的角度出发，大数据的功能定位可以依据客户、运营和产品三条主线来进行规划：客户主线主要包含客户洞察和客户体验两个方面，以客户生命周期管理的思路进行规划，包含客户画像、行为追踪、异常客户、信用评级、售后评价和障碍发掘等，目的在于全面地了解客户，帮助客户，捋顺销售流程，消除消费障碍；运营层面包含成本管理和内部效率提升，从营销、仓储物流、财务和人力资源四个角度出发，规划对应的大数据功能；最后在产品角度，主要考虑产品供应链的上下游和业绩增长方面，以渠道商、供应商和销售部门为对象，评价信誉、信息协同、风险管控和销售预测及提升，如图 1-2 所示。

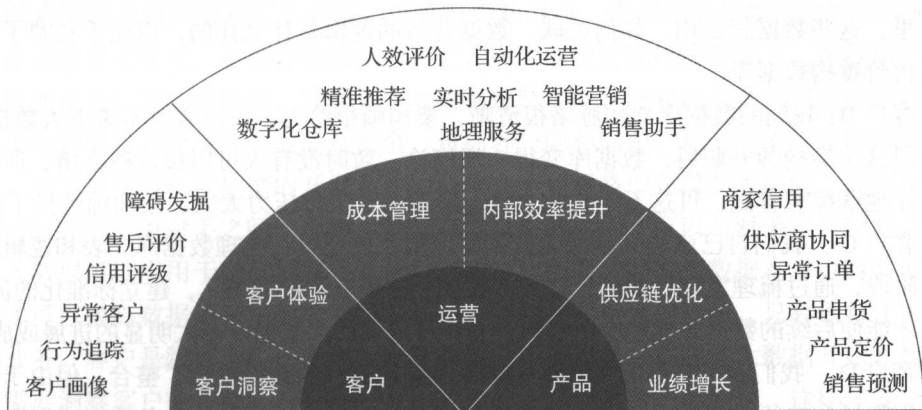


图 1-2 电商和零售企业大数据功能定位

1.2.3 启动契机

数据从无到有，甚至在最初的时候只是单纯的 Excel，逐步积累到一定规模时，即当前的工具拖延甚至对业务有重大影响，企业开始选择新的工具，直到下一个阶段，继续周而复始，但始终没有选择专业的大量数据处理工具，这就与企业的发展阶段相关。

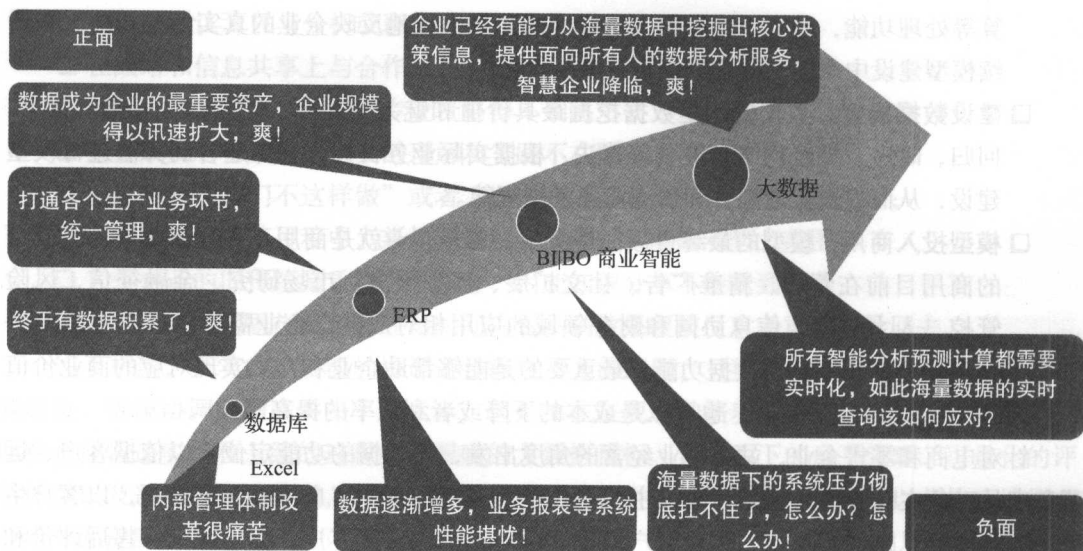


图 1-3 企业级大数据之路

然而大数据在什么时候开始启动最合适呢？我们在工作过程中接触的客户遇到的问题大体可以分为以下四种：

- 客户 A：之前我们的数据不多，所以不需要大数据，有 SQL Server 就够用了，但当我们准备使用更多数据帮助业务时发现，我们已经理不清楚到底有多少数据，都存在哪里，这些数据怎么用，如何关联，数据背后的逻辑是什么样的，以至于花费了大量的代价重构数据库。
- 客户 B：我们的数据库架构脉络很清晰，架构师很给力，所以暂时不考虑大数据产品，但这位架构师升职后，数据库变得一塌糊涂，暂时没有人可以接这些事情，所以只能让他继续兼着做，可这不是长久之计，最后由于工作压力太大，架构师选择了离职。
- 客户 C：我们自己已经开始做大数据了，但目前还处在梳理数据库、表和逻辑关系的阶段，通过梳理发现后期的管理成本很高，需要有统一的团队，建立标准化的流程后，才能使后续的数据开发和应用得到提升，所以目前大数据没有太明显的进展或成果。
- 客户 D：我们有了自己的大数据团队，已经把自有的数据进行了整合，但由于存在流程问题和执行力问题，团队收集到的数据总是不准确，有的与业务系统或财务系统相差甚远，有没有大数据，现状都差不多。

观望、大数据时机问题、小数据可以解决，这些都是对于大数据战略延迟的借口，对大中型企业来说，大数据应立即开始规划和实施，以避免后期付出更多不必要的成本，否则不仅会限制业务发展，影响销售业绩，而且更容易丧失市场的先机。而对于初创企业在追逐销售规模和利润的同时，在企业成本可承受范围内，可以开始考虑建设大数据雏形，这样不仅可以为后期大数据建设做铺垫，同时也能够在现有业务基础上，帮助企业找到深层次的业务

问题,改善经营环境,甚至找到潜在市场,助力企业快速发展。

1.2.4 大数据历程

大数据从哪里开始着手,大数据工作又包含哪些,如何规划和落地?

大数据的着手首先要结合企业自身的战略规划,包括企业的发展阶段、外部市场竞争环境、上下游供应商的水平以及市场信息技术的成熟程度。

1. 企业的发展阶段

企业的发展阶段决定了企业大数据的目标定位,不同的战略阶段需要有对应差异的目标策略。初创期企业的重点目标在于生存,因此这个阶段主要是奠定好数据收集、标准建立与整合的基础,更多的大数据工作通过自身招聘或者外包来实现即可。发展期企业已经有了稳定的支柱收入来源,为了持续稳定的发展,这个阶段的主要任务是业务规律的发现与风险的管控,大数据工作内容通过建立相关的业务模型,并有效地传递给各个业务部门,帮助其持续稳定目前的业务状态,及时发现风险。成熟期和衰退企业除了支柱收入之外,还有更多的其他收入来源,为了减缓或降低迈向衰退的速度,这个时期企业应该更加注重创新,包括通过现有业务创新、关联上下游产业进行创新、跨行业进行创新,因此大数据的工作重点在于打通行业数据孤岛,搭建数据平台,建立数据合作共享,跳出企业界限,通过数据价值帮助企业衍生业务内容,更大地提升效益。

2. 外部市场竞争环境

市场竞争环境包括宏观经济环境、竞争对手和客户。宏观经济环境的发展与战略引导,决定了企业的发展速度,甚至企业的生死存亡,在当前的宏观经济环境下,大数据产业还是朝着积极和正确的方向发展的,同样企业大数据也是如此,随着该项技术的愈发成熟,这个阶段对于企业来说需要更加关注信息安全,包括客户隐私问题、网络安全问题和内部流程管控问题。竞争对手对于大数据的开发和使用程度在企业战略中是需要考虑的,同质化的开发与应用可以缩短竞争对手之间的距离,但只有创新型的应用才可以与对手拉开距离甚至打败对手。大数据可以应用于产品创新、业务创新和流程创新,同时大数据本身也需要创新,只有创新才可以提高数据的使用效率,只有创新才可以挖掘更大的价值,只有创新才可以驱动更新的业务。客户是数据产生的来源,也是数据应用的对象,在制定大数据开始的战略过程中,也需要洞察客户的规模、客户的行为、客户的意愿以及客户的需求,这样才能令大数据的整体规划有效落地。

3. 上下游供应商的水平

企业大数据发展到一定阶段,为了更加高效地发展企业,必将实现上下游的效率随企业发展同步提升,因此大数据的战略会受到上下游供应商水平的影响,对于超过企业水平的供应商大数据能力,企业可以选择跟随和效仿的模式,而对于低于企业水平的供应商大数据能力,企业应该帮助其成长,完善信息开放共享的水平,这样才能有效地提升企业本身的经营效率和效果。

4. 市场信息技术的成熟程度

从2009年起大数据经历了探索期、启动期和高速发展期，目前正在逐渐步入成熟阶段（如图1-4所示），但还没有规模成熟的标准化产品进入市场，在未来的2~3年将会是大数据技术快速发展和走向成熟的关键时期。在这个阶段，企业大数据的启动正好贴合了目前市场技术的脚步，同时可以借鉴之前很多的过往参与探索企业的经验，在企业决心及目标定位清晰的情况下，可以快速地实现企业从传统型向数字化或智能化转型的过程。

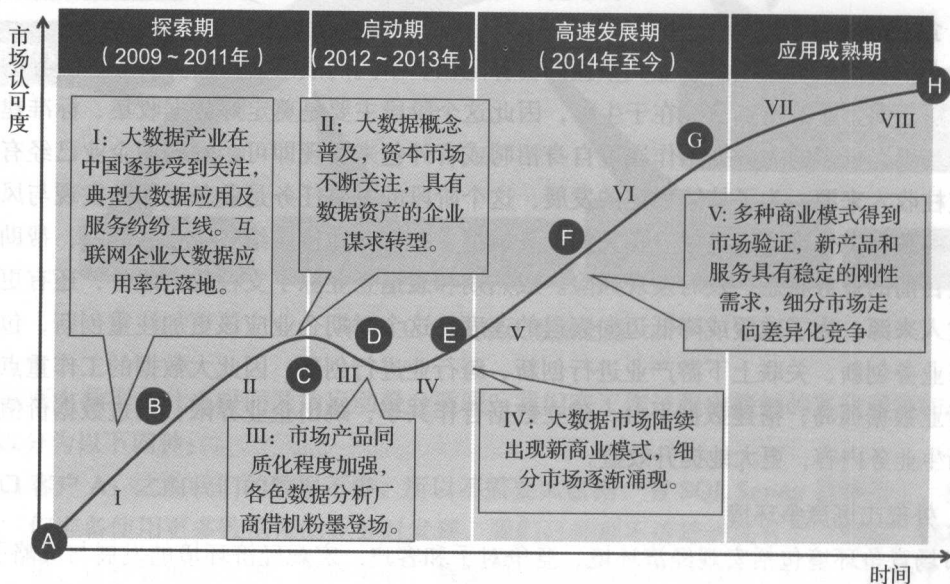


图 1-4 大数据发展阶段

为了支撑大数据战略及定位的可追溯性、可执行性和可反馈性，需要统筹考虑数据分析决策与绩效管理，实现从数据到洞察、从战略制定到业务的贯通执行（如图1-5所示）。

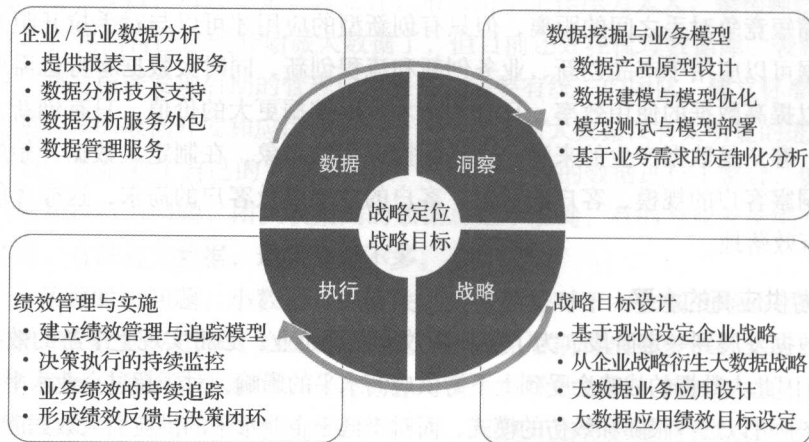


图 1-5 大数据战略规划制定闭环

企业大数据战略与其他战略相同,需要从实践中进行总结和规划,因此战略定位前,需要对企业现有数据业务进行诊断和理解,包括数据报表工具及服务、数据分析技术的支持能力、数据分析服务外包情况和数据管理服务内容,通过分析结果结合现有的数据及业务模型,找到目前业务中存在的规律及价值,同时挖掘业务执行中的实际需求,来制定大数据的发展战略,并提供对应的绩效管理与实施,督促战略的有效落地,从而令大数据战略能够真正地服务于企业,并逐步扩展范围,衍生服务至整个行业。

基于我们对整体大数据的项目理解,结合企业的大数据规划,建议采用图 1-6 所示的阶段划分与行动步骤进行大数据规划的实施。

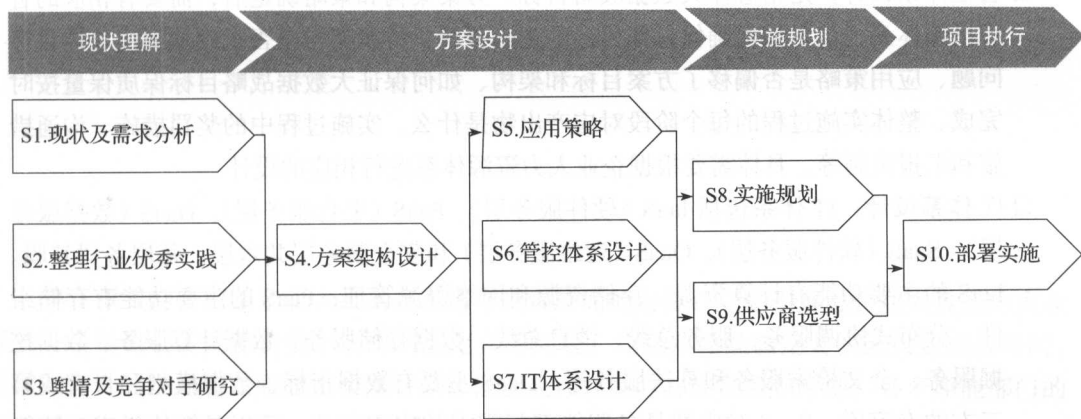


图 1-6 大数据规划实施路径

(1) 现状理解

在现状理解阶段：主要涉及企业现状及需求分析、整理行业优秀实践和舆情及竞争对手研究三个方面。

现状及需求分析：主要了解企业目前的战略方向、技术能力和数据专业能力，结合企业总体战略发展，拆解各业务中心或部门对于大数据期望实现的功能，即确定有价值的商业目标，这个过程需要经过自上而下的思想灌输，同时也需要自下而上的认同，这样才能确保最终大数据的目标能够有效落地并产生对应的价值。

整理行业优秀实践：结合企业的实际情况，借鉴和应用行业或跨行业优秀的实践成果，能够快速帮助企业实施大数据，少走弯路。目前成熟的实践包括智能报表、地理服务、市场研究、个性推荐、精准广告、风险管控、征信服务和人力资源等。

舆情及竞争对手研究：舆情即企业舆论情报，主要是指互联网上的用户对于企业品牌、产品、事件正面及负面的评价；竞争对手研究包括竞争对手舆情、价格、客户、市场、技术等方面的信息采集、分析和研究；因此这方面的网络信息数据主要通过网络爬虫来实现，致力于帮助企业了解客户、品牌、竞争对手和市场状况，有助于企业战略制定和调整实施。

（2）方案设计

方案设计阶段包含方案架构设计、应用策略、管控体系设计和 IT 体系设计四个方面。

❑ **方案架构设计：**方案总体架构包含方案设计背景、方案的商业目标、硬件架构、软件架构、业务架构、实施计划、需要的协助、风险管控、预期整体方案的实施周期、花费成本和收益。

❑ **应用策略：**在总方案架构中，属于业务架构的部分，主要是明确各个业务中心部门需要大数据完成的业务目标，该目标可以是研究和分析报告（包括内部和外部）、模型、产品、工具甚至 IT 系统操作平台等。

❑ **管控体系设计：**是指总体大数据战略目标、方案架构和策略确定后，需要有相应的管理措施体系，包含如何确保总体目标的正确分解、方案架构是否正确地解决总体目标问题、应用策略是否偏移了方案目标和架构、如何保证大数据战略目标保质保量按时完成、整体实施过程的每个阶段对应产出物是什么、实施过程中的奖罚措施、沟通措施和汇报机制等。具体需要根据企业人力资源体系进行相应的设计。

❑ **IT 体系设计：**IT 体系包括 IaaS（硬件服务层）、PaaS（平台服务层）、DaaS（数据服务层）、SaaS（软件服务层）、OaaS（运维服务层）和安全服务层共六层，每层各司其职。IaaS 的主要功能有计算资源、存储资源和网络资源管理；PaaS 的主要功能有存储组件、分布式协调服务、服务总线、消息总线、数据存储服务、数据计算服务、数据挖掘服务、全文检索服务和算法服务等；DaaS 主要有数据指标、数据模型和自有或第三方的专家库；SaaS 对应的是各职能部门需求的应用产品，可以是智能报表、智能营销平台、精准广告平台、全景式人力管理平台和财务平台等；OaaS 主要有运营和运维两层管理；安全服务层则包括物理、网络、应用和数据的安全等内容。

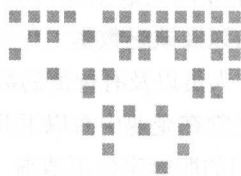
（3）实施规划

实施规划阶段包含实施规划和供应商选型两个部分，其中实施规划主要是大数据实施过程的每一个时间步骤，包含每个产品需求调研、需求讨论、产品设计、实现功能、开发节奏、产品测试、产品验收和部署上线的完整规划过程；供应商选型属于可选项目，是从企业自身的实际需求出发，寻找外部资源来加快补充企业自身人力资源和技术短板的问题。

1.3 本章小结

本章首先确定了大数据整体的战略与定位，然后明确了大数据战略从什么时候开始，从哪里开始，即大数据战略实践的过程，只有了解了这些内容，才能使得大数据真正落地。

本章需要读者重点掌握的是微观层面的知识点，这些内容直接决定了日后企业启动大数据项目时的整体方向和资源统筹。



企业大数据职能规划

第1章我们介绍了企业大数据在宏观和微观层面的定位，立足于解答企业大数据的商业模式、市场机会、延伸价值、内部功能定义等问题。当企业已经确定要实施大数据战略时，应该如何针对性地建立职能架构体系以保证企业大数据的有效实施和落地？各个职能部门的职责范畴如何定义？不同体系和部门间如何协同和流程化工作？

本章将详细讲解企业大数据职能规划体系，包括如何定义大数据部门在企业中的角色，常见的大数据职能及职责分工，不同职位的职责划分以及大数据制度和流程建设等问题。

2.1 大数据组织架构体系

要建立适合企业的大数据组织架构，首先要明确大数据部门在企业中的角色。不同的角色对应到企业内部会有不同的架构方式和职能定位。

2.1.1 大数据部门在企业中的角色

大数据部门泛指大数据中心、大数据部门、大数据组甚至是个体员工，它代表一类群体的角色扮演。按照大数据部门在企业中的不同角色和存在特征，可比喻为以下四类：路人、侍从、灯塔、先知。

1. 路人

路人是指大数据部门处于企业边缘，其存在属于可有可无的境况，这是一种危险的企业处境。

目前很多企业的大数据部门都处于这类角色中，其实质是由于企业主观上对数据不敏感、不听、不信以及缺乏数据工作文化等原因，以及客观上缺乏有效的流程和制度约束、有经验的数据工作人员以及有价值的数产出，导致大数据部门的存在与否无关紧要。

这类角色通常在企业中有以下几种行为和职能特征：

- ❑ 数据部门的职能定位不清晰，发展规划不明确，部门建设毫无方法可言；
- ❑ 缺乏有效的数据工作目标和数据价值产出；
- ❑ 数据工作从未参与企业运营落地环节，更无法渗透到企业核心业务流程；
- ❑ 数据部门缺乏“大领导”，无法直接跟企业 C-Level 的领导层进行汇报；
- ❑ 数据部门通常都是由个人或少数员工从事，甚至由运营人员兼任。



注意 对于大数据部门是否处于这种状态，通常只需回答一个问题：“如果没有大数据部门，企业会损失什么？”如果无法准确回答或含糊其辞，那么这个答案就是肯定的。

2. 侍从

侍从，即随从侍奉，这体现了大数据部门角色定位于企业辅助层面。侍从的角色相对于路人有明显的提升，该角色已经处于有明确工作需求的状态；但与此同时，大数据部门的这种状态也存在明显的问题：缺乏独立和自主性，侍从从来都不会自己决定去做什么，而是等待被分配工作和任务。同样，在企业中的大数据部门也无法决定企业在业务层面应该做什么、怎么做等问题。这种角色通常提供的职能包括如下几个方面：

(1) 数据管理

数据管理工作包括：数据配置管理、数据权限管理、用户权限管理、数据导入管理、数据导出管理。

- ❑ **数据配置管理**：主要进行数据存储、安全、排除设置、并发控制、进程控制、结构控制等管理工作。
- ❑ **数据权限管理**：主要进行数据保存、新增、删除、更新、备份、合并、拆分、导出、打印等管理工作。
- ❑ **用户权限管理**：主要进行用户新增、删除、重置、过期设置、共享、安全等管理工作。
- ❑ **数据导入管理**：主要进行数据导入格式、时间、条件、规则、异常处理、记录数、来源等管理工作。
- ❑ **数据导出管理**：主要进行数据导出格式、时间、条件、规则、记录数、加密、位置等管理工作。

(2) 数据查询

很多企业的数都在 IT 中心进行统一管理，而大数据部门也属于 IT 眼中的“业务部门”。由于大数据部门天生具有接触数据和处理数据的需求，因此很多时候也会被开放某些附属库、从属库或复制库的权限。某些情况下，大数据部门也会承担类似“取数”的功能，

这类需求在某些情况下会频繁发生，例如：

- ❑ 大型活动之后，没有数据权限的业务部门可能会发出“看结果”的需求；
- ❑ 当出现意外运营情况时，业务部门也会想要“先看看数据”；
- ❑ 做年度、季度、月度和周度等计划性的总结及规划时，业务部门也会想“参考下数据”；
- ❑ 规律性导出的日报、周报、季报、半年报、年报的详细和结果数据。



注意 限制业务部门的“取数”权限从企业宏观来讲利于数据安全把控，这是实现数据安全的途径之一。但从整体来看，如何平衡安全和工作效率，并释放人力和时间资源到更好的工作或项目机会上，需要进行权衡。毕竟，数据安全不只有权限控制这一种方法，而且只有这一种方法也无法完全保证数据安全。

（3）数据校验

这里的数据校验是指用一定的方法保证多数据源之间的完整性、一致性、准确性、及时性和有效性。

数据校验通常存在于大型企业中，这类企业往往存在多平台、多系统、多生产环境和多测试环境，此时如何保证多个系统对于同一业务主体的测量满足上述条件就要通过数据校验工作来实现。



注意 数据校验（某些公司也称为数据治理）是保证和提升数据质量的重要步骤之一，如果该过程缺乏有效执行，将很有可能导致“Rubbish in, Rubbish out”的局面，后续所有数据工作的价值将无从谈起。

（4）数据统计

大多数日常报表需要通过技术开发形成产品报表体系，以提供日常业务支持。当有突发性事件或活动时，需要人工整理和汇总报表。日常报表完成后，通过自动发送邮件或短信、在线访问、离线客户端访问等接入。

根据数据日常报表提供频率和周期不同，日常报表可分为日报、周报、月报、季报、半年报和年报。报告内容因公司需求而异，但基本框架是统计周期内企业整体、各运营环节KPI陈列、对比和简单分析，目的是通过周期性数据进行业务诊断，发现业务效果趋势和异常点，为业务优化执行提供基本支持。

根据数据日常报表支持对象在企业内部分工不同，日常报表可分为针对决策层的报表和针对执行层的报表。针对决策层的报表侧重于宏观的、整体的效果汇总和结果分析，借助对比、趋势和主要维度下钻等方式进行初步分析并定位结论和问题点；针对执行层的报表侧重

于微观的、个体的效果分析，各业务执行层只针对各自业务维度进行分析，并提供实际可行的操作型建议。



注意 对于数据指标的设定，既要包括公司核心结果指标如利润，又要包括各个业务节点的过程类或间接辅助类指标，以更全面地评估和定性整体及各业务线的工作结果。

3. 灯塔

灯塔意味着企业的工作方向或职能开展需要大数据部门进行指导，此时大数据部门承担着以下三类角色和功能：

- ❑ **剖析过去。**对过去所发生事件的原因进行剖析，找到影响全局或特殊事件的关键因素并加以提炼以形成优化或改良机制；找到数据中的频繁规则并提炼出可供现在或未来使用的业务方法；从海量数据中发现数据知识，并能通过知识来引导业务行动或进行业务优化规则的启发。
- ❑ **监控现在。**对数据实时的监控和反馈通常是大数据部门的必备职能之一，数据反馈的实时性通常对于在线活动影响极大，无论是基于预测的、异常波动区间的还是数据分布模型的监控方法，只要能快速、有效并且准确地告知业务主体当前发生的问题，并配合业务一起剖析问题，尽快解决类似于流量作弊、黄牛订单、恶意注册、虚假投资、骗保等问题，能为企业节省大量时间、资源和项目等成本支出项。很多时候，时间就是机会，而时间也是最大的成本。
- ❑ **预测未来。**基于历史情况对未来的事件预测意味着业务在开展行动之前需要有明确的目标导向，基于目标可以制定明确的 KPI、匹配为实现目标所需要的资源、预估行动成本和收益、平衡不同项目的机会成本和对企业整体战略布局的影响。



注意 大多数企业中的大数据部门都有类似于数据挖掘、数据分析、专项分析类的职责，这类工作的核心价值通常不是产生多少模型、几种算法、多少报告等，而是直接对于企业整体销售和利润的提升，或在保持相同销售和利润水平下对成本的控制和缩减。当然，某些企业内部会由于各种原因，比较注重知识产权、专利申请、科学研究、学术报告和期刊等的影响力，这些视具体情况而定。

这类角色通常通过一定的模型、算法、流程和机制对数据进行解析，大多数的工作都是通过专项数据挖掘或分析的形式开展。

数据专项挖掘分析是指针对某一特定课题或需求，采用专项分析或长期课题分析的形式对数据进行深入挖掘和分析，以提炼出相应结果或方法论供业务参考或使用。数据专项挖掘分析是数据发挥价值的重要手段，更是数据辅助支持作用的关键，大多数公司的数据工作意

义都来源于此。

为了提高数据工作的针对性,数据专项挖掘通常按业务模块划分,常见的数据专项挖掘分析模块包括市场分析、营销分析、运营分析、会员分析、用户体验分析、销售分析、移动分析、O2O 分析、库存分析、供应链分析等。不同分析模块课题依业务需求而定。

4. 先知

在上述三类角色中,我们讨论的知识前提都是数据依托于业务主体开展工作。但无论开展的工作是预测性的、剖析性的还是知识挖掘性的,可以说没有业务就没有数据发挥作用的土壤,更无法落地应用和实施。因此,从某种程度上看,数据是一定要依托于业务主体而存在。那么数据真的只能处于依托作用或依托于业务而存在吗?

在大数据时代的当下,身边所有介质所产生的任何属性、行为、结果等都可以通过一定的形式进行记录。现在除了传统的结构化数据外,还包括半结构化和非结构化的数据形式或类别,例如日志、文本、视频、语音、图片、文档、XML、HTML 等。这些数据形式或状态可以被人类识别并加以有效分析、整合和利用,既然人类可以做到,那么理论上在一定条件下计算机也有机会这样开展工作。

人类开展工作的前提是从出生开始便不断接收外界各种信息源的刺激和学习,相对的,计算机所能接收到的信息相对于人类接收到的数据和信号而言,都是碎片化并且微乎其微的。基于计算机视觉、模式识别、自然语言处理、机器学习、深度学习等领域的人工智能正在被人们进行广泛的研究。假如通过一定途径将人类接收到的所有信息都能传递给计算机,那么计算机便可识别、加工、分析、应用和预测这些信号。因此,解决了这些问题之后,计算机智能便可脱离业务主体而存在,甚至在一定程度上,它可以创造业务、思考业务和优化业务并找到最优化方法进行求解。

目前,这类角色在企业和社会中还没有大规模的综合性应用案例,但在很多垂直领域中已经有所突破,例如机器翻译、语音识别、图片识别、自动规划、智能无人汽车、智能博弈等;而在学术和知识研究领域也有各自阵地,包括深度学习、神经网络、机器学习等。未来,数据的价值将借助于传感器、海量数据、数据推演的模型和算法、自动程序设计、自动控制以及硬件集成等方式独立开展行动。

2.1.2 常见的大数据职能及职责

常见的大数据组织架构分为四种类型,根据不同公司的性质可分为分散型架构、集中型架构、复合型架构和矩阵型架构。

1 分散型架构

在分散型数据架构中,数据作为单独的部门位于各个业务中心之下,职责是提供本中心的数据支持。如图 2-1 所示,营销中心、运营中心、会员中心和 IT 中心都有自己的数据部门,各个部门相互独立。

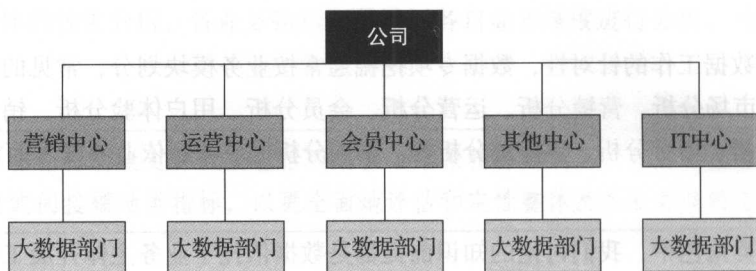


图 2-1 分散型数据架构图

分散型数据架构常见于企业创建数据体系的初期，初衷是先将数据置于某个中心之下，待数据工作正常开展并卓有成效之后，再在其他部门成立数据部门并辅助业务工作。

分散型数据架构下，各大数据部门的职责是高度相似的，包括：

- 运营业务数据统计；
- 用户体验、SEO、用户研究等通用方向的分析；
- 各自业务中心业务活动效果分析；
- 关键业务项目的数据挖掘和分析；
- 数据报表和数据产品开发（主要是 IT 中心的大数据部门）；
- 机器学习算法实现和集成（主要是 IT 中心的大数据部门）。

这种数据架构的优势非常明显：前期投入较小，只需人员成本和极少的系统成本便可开展工作；数据从业人员由于处于业务工作体系内，对业务熟悉度较高，数据落地价值更大；另外，相同体系下的各个部门协同工作效率更高，利于业务方数据理解和执行。当然，这种架构的缺点也是显而易见的：

- **数据质量难以保证。**各部门数据来源分散且不完整，数据质量难以保证，基于未知质量上的数据结论可能无法立足。
- **数据共享困难。**不同数据部门之间的数据孤立还会导致数据孤岛的出现，不同的思维方法、工作机制，甚至定义方法不同导致数据源和数据结果无法流通、共享和综合应用。比如，对于转化率的定义方法，可能有订单 /UV、订单 / 访问、订单客户 /UV 甚至件数 /PV。数据共享困难一方面可造成数据价值难以最大化传播，另一方面在同一个数据项目的处理上也造成重复的人力、时间和物力投入并导致资源浪费。
- **数据结果混乱。**由于数据来源不一致或同一来源下定义口径的不同，各个业务部门汇报结果可能存在数据出入。这会影响决策层对业务结果的判断，同时影响数据的可信度。
- **难以形成合力。**各部门基于自身需求搭建支持体系，不同部门间难以形成合力共同搭建对全公司服务的数据支撑点。

2. 集中型架构

集中型数据架构与分散型数据架构相反，它是把所有的数据工作汇总到一个中心集中统

筹规则，通常该中心是信息技术中心或 IT 中心。图 2-2 为典型的集中型数据架构图。

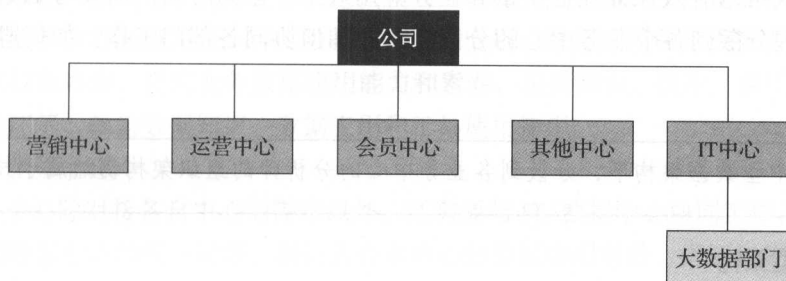


图 2-2 集中型数据架构图

该架构下由于所有的数据工作都集中到 IT 中心，因此大数据部门工作职能高度集中，主要包括：

- 异构数据和主从数据的校验；
- 数据统一管理和权限管理；
- 数据报表开发和产品开发；
- 根据业务需求的数据抽取；
- 机器学习算法实现和集成；
- 针对各业务线的数据分析。

这种数据架构体系有效地解决了数据源不一致和数据口径定义的问题。由于所有数据从生产到应用都由该中心统一负责，数据质量度较高。这种数据架构的主要问题是业务理解与支持较弱：


- **业务工作流程复杂。**所有业务中心的数据需求都需要经过该中心处理，需求沟通、确认、实施、反馈的流程较为复杂，影响业务对数据需求的积极性与主动性。
- **业务理解度不够。**在该中心统筹下的数据体系，附带了技术的思维方式和工作方式，对业务的理解程度低，使得数据难以落地应用。
- **技术响应及时性差。**该中心的部门都有各自的工作计划和排期，业务方多而杂的临时需求影响其正常工作，大量需求可能被积压甚至无限延期。

为了解决集中型数据架构带来的业务应用问题，行之有效的一种方法是派驻数据分析师入驻到各个业务中心。这能在很大程度上缓解技术类中心“不懂业务”的被动局面，但对数据分析师个人素质和能力有较高要求：

- **扎实的基本数据素质。**分析师需要具有扎实的基本数据素质，能及时、有效、准确地解答业务数据问题。
- **良好的个人时间把控能力。**由于身处业务中间，分析师会面临很多临时需求，包括咨询、取数、分析、报告等，这就要求分析师具有良好的个人时间管理素质。
- **完善的工作流程和机制。**流程和机制可以使各项工作有据可依，过滤无效需求的同时

保证数据安全性、有效性、及时性和落地应用价值。

上述方式可以有效保证数据质量和业务应用效果，但同时我们需要考虑数据之外的问题：如何管理分散到各个业务中心的分散人员？如何协同各部门工作？如何避免交叉管理问题？

 **注意** 在集中型数据架构下，分散到各业务中心的分析师的组织架构仍然属于技术中心。

3. 复合型架构

复合型数据架构是建立在分散和集中基础上的复合组织架构。数据端集中到统一中心之下管理，该中心通常是 IT 或数据中心；业务端分散到各业务中心之下设立数据支持部门，如图 2-3 所示。

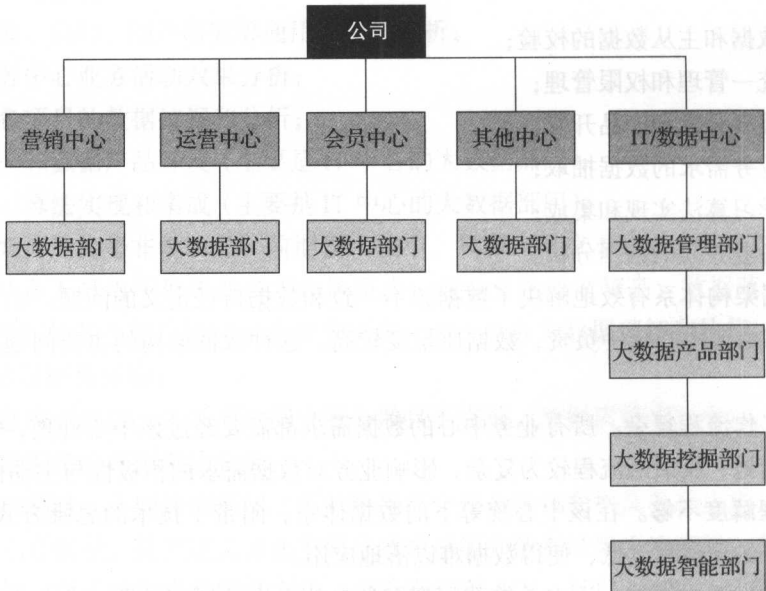


图 2-3 复合型数据架构图

复合型数据架构既能保证数据的质量标准化，又能保证各个业务节点的数据落地应用，同时还可以结合各业务共同需求以及公司战略发展需求开发全局应用的智能产品。不同中心间的分工如下：

(1) IT/ 数据中心

IT/ 数据中心的数据职能是对接全公司所有业务高级需求，统筹整体并进行相关数据产品开发：

- 统一口径。数据源的定义、数据出口和抽取逻辑的统一、数据指标和应用场景的规范等。

- 搭建平台。经过整合和清洗的干净的数据源甚至数据平台、报表可视化等。
- 智能数据产品开发。自动化数据挖掘模型封装和开发、BI、个性化推荐等。
- 对接业务中心高级需求。深度数据源抽取和应用、数据建模和挖掘技术支持等。
- 数据技能培训。提高业务数据应用能力和素养，包括知识、技能、素质、最佳实践场景推广等，涵盖数据知识、数据应用和工具使用知识。

(2) 各业务中心

各业务中心除对接各自中心的需求以外，还需要与 IT/ 数据中心协同工作：

- 根据数据中心的统一规范，制订适合本中心的数据应用场景、指标和分析体系等；
- 收集各自中心的零散需求并反馈到 IT/ 数据中心，参与 IT/ 数据中心公司级数据产品开发和应用，参与环节包括底层收集、数据 ETL、数据建模、数据可视化、数据智能应用等——该项工作是数据协同工作的重要产出。

4. 矩阵型架构

矩阵型数据结构常见于第三方服务或外部服务公司，属于项目管理类企业的常见架构，对于这种企业而言，项目制的工作方式是企业业务运作的基本模式，如图 2-4 所示。

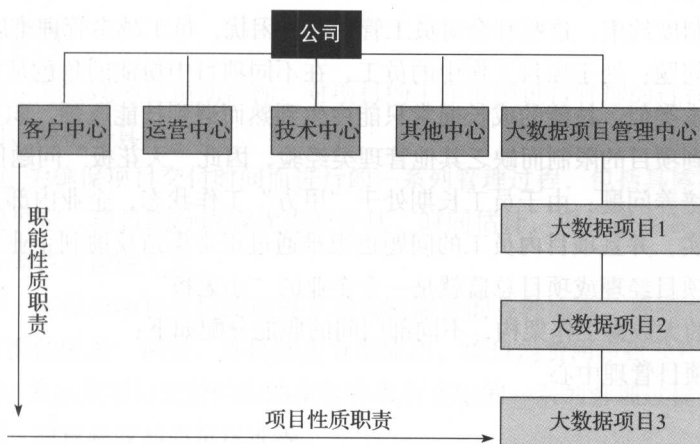


图 2-4 矩阵型数据结构图

这种模式或职能结构具有以下特点：

- 所有大数据项目都有直接项目负责人，该角色可能是项目经理也可能是项目总监，具体视项目重要性而定。
- 不同项目间通常相互独立，可独立核算成本和利润，这使得所有项目可衡量、可优化和可改进。
- 业务动作以项目为导向，除了企业管理类部门外，其他所有的职能部门都是为项目提供服务，支持项目工作的有效开展。
- 公共资源池的有效利用：项目间的资源利用可以从公司整体统一调度，通常以设立资

源池作为调用出口，所有项目资源（人力、设备、技术、产品等）使用完成之后可快速回收并调配到其他资源中。

❑ 项目间的资源流通性提升：不同项目间虽然独立运营并参与核算，但资源也可以互通使用，这会利于保证有效资源的有效调节和最大化使用。

❑ 大数据项目管理中心统一协调：作为所有项目管理的枢纽，该角色承担的项目工作包括项目获取、组建、管理、运营、重组、回收、调节等，整体把控性更强。

这种整体与局部的有效统一使得所有的大数据工作环境都相对可控，利于企业利益最大化，但同时也存在一些不可避免的问题：

❑ 员工缺乏归属感：大多数参与项目工作的员工，通常需要驻扎在客户阵地前线，这使得项目完成之后员工需要根据下一项目需求调度到其他项目中重新投入工作。很多情况下可能会到全国各地做项目，导致员工很难产生归属感，容易造成员工流失。

❑ 企业人员有效管理问题：对于大多数项目工作制的劳动方式而言，大多数时候都在“甲方”工作，这使得员工的“工作过程”很难把控，因此大多数项目员工会以项目交付成果作为考核依据；除此之外关于员工的费用、社保、培训、晋升、汇报、福利、知识等所有问题由于缺乏有效的基于地理位置的管控，只能通过在线系统开展并需要依靠公司制度约束，这些都会对员工管理造成困扰，员工越多管理难度越大。

❑ 员工成长问题：处于项目工作中的员工，在不同项目中扮演的角色是类似的，应用的技能也基本类似。技能的成长通常只能由生到熟而遇到技能瓶颈，职业通道和发展路径上又受到项目的限制而缺乏其他管理类经验，因此“天花板”问题比较突出。

❑ 企业文化培养问题：由于员工长期处于“甲方”工作状态，企业内部工作文化很难进行有效落实，并且项目内员工的问题也很难通过正常渠道反馈到企业工作流程和机制中，通常项目经理或项目总监就是一个企业的“小老板”。

对于矩阵型的大数据工作架构，不同部门间的职能分配如下：

（1）大数据项目管理中心

核心职能：

❑ 资源管理：根据公司项目开展需要，建立和健全项目资源管理制度，实现公司所有资源在项目内的总体协调与调度最优化，以保证资源效率最优、利润率最大。

❑ 项目管理：组织和策划公司项目招标、计划实施与协调，确保各项目的有效推进和落地。

❑ 质量管理：制定施工方案、质量工作标准和验收标准，组织质量管理培训、逐步推进项目活动全过程的质量管理工作。

❑ 费用管理：组织实施工程项目管理的项目经理责任制和项目成本核算管理。

❑ 监察管理：对各项目中可能存在的影响公司整体利益的外包项目分派、内部资源的外部利用、项目违规操作、个人边缘利益以及其他违反公司规章和制度的监督和管理措施。

非核心职能:

- ❑ 知识管理: 针对项目实施过程中遇到和应用的场景、行业、案例、模型等知识物料进行统一汇总和管理, 形成可供企业所有项目参考的知识库, 最终根据企业市场形态建立针对性的解决方案。
- ❑ 培训管理: 项目招投标、实施、验收等过程中所需的各种技能和职业素养要求的培训, 重点在于满足项目工作需求, 是对普通职业技能的拓展。
- ❑ 人员管理: 项目工作中所需人员的管理, 包括组织规划、人员选聘和项目核心骨干建设等一系列工作。
- ❑ 档案管理: 建立和完善项目信息、档案信息制度, 组织和指导建档工作并及时汇总和更新档案信息。



注意 大数据项目管理中心和各项目中心的职能中, 除资源管理和项目管理外, 其他职能可能会根据公司实际运营情况有所差异, 某些公司甚至会采用各项目组独立核算成本的方式。

(2) 各项目中心

- ❑ 范围管理: 为实现项目预期目标, 对项目的工作范围进行管理的过程, 包括范围的界定、规划、调整等具体工作。
- ❑ 时间管理: 为确保项目交付时间而进行的一系列管理过程, 包括具体项目实施的规划, 实施过程界定, 项目细分内容优先级评估、时间估计, 项目进度控制, 周期性监察, 进度报告等各项管理工作。
- ❑ 成本管理: 在保障项目交付的前提下对实际需要的各种成本、费用的管理过程, 包括软硬件资源的配置、调整, 弹性解决方案应用, 项目内费用审批及控制等各项工作。
- ❑ 质量管理: 为达到项目交付约定的质量要求所实施的一系列管理过程, 包括质量规划、质量控制、质量验收和质量保证等。
- ❑ 人力资源管理: 项目内的人力资源管理通常是对于项目内部人员的工作职责、范围的调整, 以及为最大化人力资源产出而实施的工作时间、效率和结果的一系列管理措施。
- ❑ 风险管理: 对项目工作过程中涉及的可能会影响项目交付时间、交付质量、交付数量等交付成果的各种不确定因素的识别、量化、规避和控制等管理措施。



注意 很多公司为了避免项目的失控并保证公司利益最大化, 都会设置项目内的双管理(两个项目负责人)检查的制度, 这样不但可以保证各利益方相互监督, 同时又能最大限度地避免利益主体抱团。

2.2 大数据职位构建体系

在团队组建过程中，科学地定义职位体系直接影响到大数据实施的效率和质量，由于大数据的创新性和严谨性，会有一批新的岗位，例如首席数据官、大数据解决方案架构师、大数据采集工程师，大数据研究员等；同时，也会强化原有岗位的新生命力，例如网络工程师、算法工程师、系统架构师、咨询顾问、数据库管理与开发等。整个职位架构体系，如图 2-5 所示。

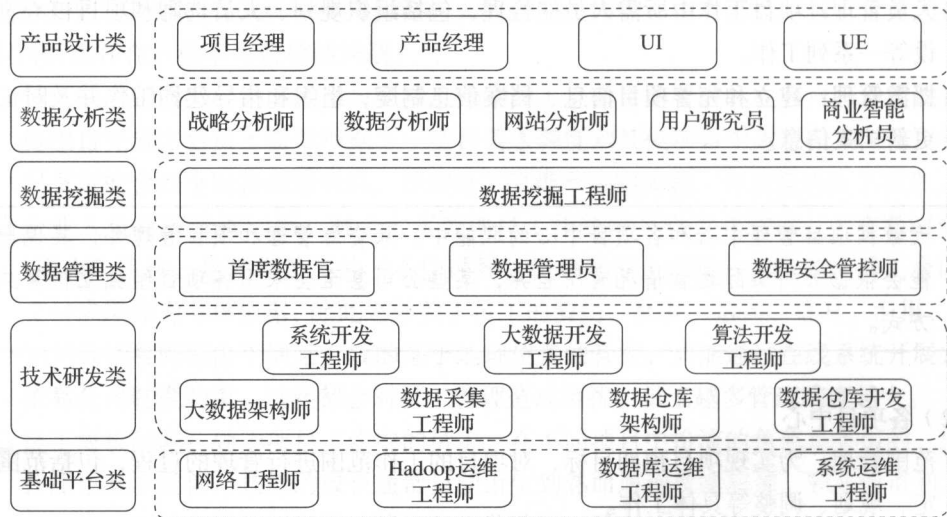


图 2-5 大数据职位构建体系

2.2.1 基础平台类

大数据基础平台共分为硬件平台和软件平台两大类，硬件平台包括服务器、操作系统和网络维护等工作，主要由网络工程师负责；软件平台包括 Hadoop 运维、数据仓库管理、软件系统运维等工作，由 Hadoop 运维工程师、数据仓库管理员和系统管理员负责。

1. 网络工程师

网络工程师在计算机行业是一个非常“古老”的职业，主要目的是维护多台计算机及其外部设备，保障计算机在资源共享和信息高速传递的稳定性。在大数据环境下，由于大数据系统是多台（几十、几百甚至成千上万台）服务器分布式环境，并且具有并行计算、实时传输的特性，对网络传输、安全、读写效率及并发的要求更高，其中共涉及多项十分重要且复杂繁琐的问题：机房网络安全问题、跨机房服务器集群之间网络传输问题、开发人员操作失误风险规避问题、跨机房硬件稳定性保障问题、磁盘高速读写问题、跨局域网的分布式集群传输效率问题、分布式集群服务器 IP 及网络配置问题。

随着大数据技术、IPv6 标准、物联网、移动互联网技术的快速发展，使得对于新型网络工程师的人才和技能要求也越来越多，由于上述每项技术对网络和硬件的要求侧重点都不一

样,也因此而细分出多个发展方向,对相应的技能要求的侧重点也不同,例如网络安全类、数据存储类、架构设计类、移动网络类、网络效率类等。

2. Hadoop 运维工程师

作为大数据产业下的一个新型职位,Hadoop 运维工程师是大数据系统稳定运行最重要的保障,其主要职责是维护高性能的 Hadoop 分布式数据存储系统,并为业务系统提供稳定的数据访问服务,开发新接口和维护原有系统。

由于 Hadoop 技术生态组件绝大部分是由 Java 语言实现并且基于 Linux 操作系统运行的,对于该职位来说,首要要精通 Java 编程和 Linux 操作系统;其次要精通 Map/Reduce 运行机制、Hadoop 集群的硬件资源(CPU、内存、存储)配置与管理、Hadoop 各组件(如 HDFS、Hive、HBase、Impala 等)的运行原理、集群组件监控、集群节点故障解决方案等;另外还需要懂得如何保证数据安全、数据存储效率、计算效率、运维效率的优化与提高等。如果觉得 Hadoop 集群的性能差时,绝大部分责任是运维工程师对 Hadoop 的了解太少,Hadoop 的效率没有被发挥出来。

3. 数据库运维工程师

数据库运维工程师在公司发展的不同阶段有着不同的职责与定位。工作内容包括负责数据库的运营和维护,包括数据库的安装、监控、备份、恢复等基本工作,并需要覆盖产品从需求设计、测试到交付上线的整个生命周期,在此过程中不仅要负责数据库管理系统的搭建和运维,更要参与到前期的数据库设计、中期的数据库测试和后期的数据库容量管理和性能优化。

当企业数据仓库中的数据量达到了一定量级时,对数据源的监控和保障数据仓库的稳定性是一个必要的工作,传统的企业级数据仓库一般都会建立在 MySQL、Oracle 等结构化数据库之上,数据库运维工程师为了提高数据库工具和服务的有效性,会选择合适的软件和硬件工具,并使用各种工具实时监控数据库性能和数据录入程序,管理数据安全和隐私并创建和分配新的数据库,从执行层面优化数据库性能提高查询和处理能力,建立数据备份、数据库故障排除和恢复机制确保信息得到保护和备份。在企业级大数据生态体系下,Hadoop 数据仓库成了数据处理挖掘的主力军,所以下一代数据库运维工程师还需要精通各种 Hadoop 技术生态组件的存储运行机制和执行管理命令(如 Map/Reduce、Python、Scala、Java 等)。最后,数据库运维工程师特别重要的一个工作是确保所有的数据符合法律规定,对整体数据质量要定期做完整的报告并反馈给工作团队。

4. 系统运维工程师

和企业传统的系统运维工程师一样,大数据系统运维工程师需要维护所有业务系统的功能运行,并监测系统的所有功能是否正常,维持系统现状,协助解决新的和现有的系统问题形成系统运维自动化流程。在业务系统进行测试和升级过程中,实现所有的目标,保持对所有系统升级之后的新技术维护,并确定系统运维的长期目标,根据运行的实际情况改进维护策略。除以上“已发生”类的问题之外,对“未发生”类的问题要及时监测,分析所有系统

的升级和应用程序，确保遵守所有计划要求，设计新上线的业务系统解决方案。监测和分析业务系统的运行指标，并保持最佳性能，与管理人员和社区成员协调落实各项业务活动，并确定所有服务器配置。最后，由于大数据生态体系的技术组件更新升级频率非常快，且新技术发展异常迅速，要求所有大数据系统运维工程师必须拥有非常高的行业知识更新和学习能力。

2.2.2 数据管理类

数据管理类岗位中的管理所针对的对象是数据，而非职能岗位中的管理角色。数据管理类岗位包括首席数据官、数据管理员和数据安全管控师。

1. 首席数据官

随着大数据的不断发展，企业对于数据的依赖也越来越强，当企业内部的数据类型和数据用途越来越多时，需要一个“数据管理者”从数据全场景和流程方面进行整体规划和管理，一个大数据新的工作岗位应运而生——首席数据官（CDO）。

该职位的职责包括：与数据所有者和数据管理员共同为内部和外部创建数据管理策略并且实现数据的准确性和制定工作流程的需求目标，定义大数据战略，设计多阶段实施路线图。建立数据管理策略及标准，创建并领导数据管理团队，监管组织内的数据质量工作，配合 CIO/CTO 和 IT 部门协同管理和完善数据管理策略的实施，并负责企业信息数据管理的预算和数据相关系统的审核。

另外，作为技术高管需要有团队成员梯队建设能力，带领团队技术探索不断创新，推进企业技术升级、技术架构完善以及数据仓库和商业智能解决方案的发展，协助业务开发团队提供售前活动和招标书，帮助评估和计划项目，协同 CIO/CTO 管理技术和项目团队。

2. 数据管理员

对于一般的公司来说，数据管理员的工作可能由数据库运维工程师来兼任，从部署操作系统，到数据库安装、设计和部署监控，防止漏洞和攻击、主数据管理、权限管理与审核等，而大数据环境下对数据管理员工作的技能要求更加精细。

对数据管理员的岗位要求包括指定并实施数据管理策略、协调和执行数据管理解决方案、数据库权限管控三大方面，所以该项职位的主要任务是：了解企业内部的数据需求信息，并将其传送给数据团队的其他成员，深入了解数据现状并收集相关资料；引导业务指标的制定和记录，协助数据分析师分析现有的报告并确定整合指标，上报首席数据官，参与制定数据管理与实施计划，指导数据库需求文件的准备；在数据管理计划实施的过程中，担任 ETL 和系统开发工程师的顾问，协助数据分析师评估任务，分析现有的报告，并帮助识别潜在的数据来源和数据库等；在数据管理与实施计划落实完成后，保障公司核心业务实体数据（例如客户、供应商、商品、组织单位、员工、合作伙伴、位置信息等）的一致性、实时性和精确性，成为企业内部的数据“交通枢纽”。

3. 数据安全管控师

数据安全是互联网行业中的一个永恒的话题,无论是对于单位还是个人,数据安全都是至关重要的,如果数据不慎丢失或者泄露,都会造成重大影响。

一般情况下企业的数据安全管控由数据运维工程师或者数据管理员负责,由于计算机和互联网的影响越来越大,人们日常生活中的网购数据、通信数据、身份信息数据不断膨胀,同时公司企业的主数据、业务数据、销售数据、财务数据也在不断增长,所以数据的安全性就越发重要,在互联网大数据时代需要数据安全管控师对系统数据安全进行严格规范和控制。

影响数据安全的因素包括计算机硬件损坏、工程师的操作失误、黑客入侵、病毒感染、企业内部发生的数据盗窃等,数据安全管控师的任务,是通过各种安全策略和安全防范手段,在这些问题发生之前制定良好的安全方案防患于未然,主要安全策略包括:协助首席数据官制定规则加密电子文档数据内容、细化数据权限控制读写删除操作、制定数据备份流程机制、制定组织结构成员数据权限关联及分级授权机制、制定数据及文档集中管理与分发规范、制定数据通信安全规范、制定数据仓库访问和操作权限等。

2.2.3 技术研发类

技术研发类岗位指的是针对大数据相关系统、软件、产品和功能进行的开发,而非 IT 系统的开发。由于大数据类的开发是一个相对完整的工作链,并且具有特殊应用需求和场景特征,因此涵盖了几乎与 IT 系统相同的职能岗位。技术研发类岗位包括大数据架构师、数据仓库架构师、大数据开发工程师、数据采集工程师、数据仓库开发工程师、系统开发工程师、算法开发工程师。

1. 大数据架构师

作为大数据技术平台成功落地的重要保障,大数据架构师在大数据技术发展之初就已经奠定了必不可少的角色基础,该职位主要负责 Hadoop 技术解决方案的整个生命周期的解决方案确定并进行引导,包括:大数据需求分析、平台选择、技术架构设计、应用设计和开发、应用测试和部署等大数据实施全流程的跟踪,并在实施过程中带领技术团队,为设计和开发大规模集群的数据处理系统提供技术和管理。

由于角色的重要性,通常情况下该职位应该拥有重点院校计算机相关专业的硕士及以上学历且至少 5 年以上 Java 编程经验,精通 Java 原理和 Hadoop、Hive、HBase、Impala、Spark 等大数据技术生态体系,熟悉常用的数据挖掘算法,如逻辑回归、决策树、关联规则、序列模式、时间序列、SVM、贝叶斯、聚类等,以便做更好的架构方案选型。除以上技术要求外,该岗位聚焦于互联网涉及的各领域平台架构设计,可能会涉及电商平台、虚拟化、云计算、数据分析挖掘等。

作为一个或多个领域的系统架构专家,更要面向未来:设计领先的软件架构,洞察所在领域的系统技术发展趋势,提出新的系统架构理念,主导架构技术项目开展架构原型的验证,保证未来新产品的软件架构具有领先的架构竞争力;改进已有产品的软件架构,分析行业内重

点产品的软件架构，识别软件架构设计方面的问题，提出解决建议和方案，并指导改进；提升团队的软件架构设计能力，时刻洞察技术发展动态，指导技术开发人员及时升级系统技术。

2. 数据仓库架构师

数据仓库的开发和管理在大数据时代显得尤为重要，相关的数据库管理、运维和开发技术，将成为广大 BI、大型企业和咨询分析机构特别看重的技能体现。而之前一般企业中的数据仓库架构师都由数据部门开发经理兼任，同时数据仓库团队工作内容比较纯粹，所以该职位可视数据仓库量级和企业实际情况而定，由其他职位兼任或单独设立都可。

数据仓库架构师的主要责任有：数据仓库的架构设计及数据集市建设，带领团队落地及后续的运维，负责各条业务线的数据整合方案设计及日志规范，数据分析指标体系建设及元数据管理，并要稽查和监控数据质量，数据报表系统及相关数据产品的研发和数据需求的沟通及数据开发项目管理。

在技能要求上，精通 SOL、SSIS、SSRS 和 OLAP 等进行数据库及数据模型设计，如使用 Oracle/HANA 建立数据仓库，熟悉 Kettle、Informatic、Datastage、DataService 等 ETL 开发工具（目前很多 ETL 工具也支持 Hadoop），了解行业内的各种数据仓库应用案例和商业智能（BI）实时动态。如使用 Hadoop、Storm、Spark 建立数据仓库，精通大数据分布式平台技术，熟悉 Java、Scala、Map/Reduce、HiveSQL、SparkSQL 等技术。同时，根据企业数据仓库技术发展的实际情况，可能需要使用 Oracle 与 Hadoop 相结合的方式完成工作。

3. 大数据开发工程师

大数据相关的技术组件包括分布式存储（结构化与非结构化）、缓存、查询、计算（实时与离线）、监控与管理、资源调度等，为了保障各技术开发的专业性，一般以开发工作的内容进行划分：Hadoop 开发工程师（离线计算）、实时计算工程师、数据处理工程师、文本挖掘工程师（非结构化数据处理）等。

Hadoop 开发工程师需要精通包括：HDFS、HBase、Hive、Impala、Zookeeper、YARN、Map/Reduce 等在内的所有组件部署、调优与开发。Hadoop 技术应用广泛，开发过程中还会涉及 Hadoop 版本的快速迭代升级，需要和 Hadoop 运维工程师协同开展工作。

实时计算所涉及的技术包括 Spark、Storm 两大核心组件，而 Spark 与 Storm 组件的开发语言都各自不尽相同，这无疑大大增加了实时计算工程师的开发难度，除了精通 Java 之外，还必须精通 Scala（Spark 是由 Scala 写成）、SparkSQL 和 SparkStreaming。

以上技术都是针对结构化和半结构化数据的开发处理，非结构化数据的开发处理一直都是相对更繁琐的工作。比如，文本挖掘工程师的工作是对非结构化数据进行抽取、解析、建立全文索引等，使非结构化数据转化为有价值的结构化或半结构化数据。数据处理工程师主要负责分布式存储与计算平台中的数据处理与传输，承担着“数据搬运工”的角色，不管是结构化或半结构化数据还是非结构化数据，一般都会使用到 Kafka 或 MQ 等组件进行数据的解析与传输。

4. 数据采集工程师

数据采集工程师的主要职责是收集和处理海量原始数据，工作内容包括：脚本编写、网页获取、调用 APIs、编写 SQL 查询等。

由于数据源的存储及展现方式不同，数据采集分为外部数据采集和内部数据采集，外部数据采集通常指的是互联网网页采集（也称网络爬虫），工作任务是通过搜索引擎网络爬虫相关技术和正则表达式，从抓取下来的 HTML 页面数据中提取网页数据信息，这要求工程师必须精通互联网内容搜索产品（例如百度、谷歌）的设计和架构，熟悉搜索引擎、互联网网页及反爬虫技术的工作原理，熟悉 Linux 操作系统，具备搜索引擎开发的研究能力，使用到的开源技术工具有：Nutch、Heritrix、larbin、HtmlParse、Scrapy、Lucene 等。

内部数据采集是指存储在企业内部数据系统（如 Oracle、MySQL、NoSQL、Log 日志）中的主数据 / 业务数据和企业网站 / App 端中用户行为数据的采集。企业内部数据采集的工作任务是通过数据库抽取相关技术（Java、Sqoop、GoldenGate、Canal）把存储在企业数据库系统中的数据抽取出来，重新整合、同步与存储；企业网站 / App 数据采集是通过 JS/SDK 等技术手段，把网页 / App 端的用户登录、点击、查看等行为收集起来，同步到后端的数据存储系统中。

通过内部、外部数据采集到的数据最终都会存到分布式文件系统（Hadoop、Spark）中统一存储，便于后续的数据分析与挖掘。这些工作要求工程师了解企业数据流通机制，精通 Oracle、MySQL、NoSQL 等数据库的工作原理和主流的大数据接入技术（Kafka、Storm、Flume、MQ、SparkStreaming），熟悉 Nginx 日志、算法设计、数据结构、Java 和 Scala 等。

5. 数据仓库开发工程师

传统数据仓库开发团队在企业技术岗位中属于不太容易看到“效果”的团队之一，而且所需的人数不太多。但是在进入大数据时代，代表着更多类型（尤其是非结构化类型）的海量数据不断涌现，客观上要求对数据进行实时采集、分析和传输，这就对基础设施性能提出了严峻挑战，尤其是对运维管理者数据仓库开发和管理人员提出了更高的要求。

数据仓库开发工程师除了需要基于 Oracle/HANA 开发外，还要基于三大不同类型的数据库进行应用开发：分布式数据库 NoSQL、Hadoop 体系，分布式数据库 HBase/Hive 和实时分布式计算框架 Spark/Storm。由于 NoSQL、Hadoop、实时计算技术可供使用的 ETL 工具比较少，所以目前企业数据仓库开发工程师和大数据开发工程师使用到的大部分技术是相同的，但数据仓库开发工程师的工作更侧重于数据层设计与开发、ETL 流程开发和优化，完成结构层次合理、灵活可扩展的数据仓库结构。同时，这些工作也都需要对 Hadoop、NoSQL、实时计算技术有深刻理解且对业务精通的人才能胜任。

6. 系统开发工程师

大数据系统按应用类型分为数据可视化类与数据应用类。

可视化类系统包括：商业智能、数据监测、舆情监控、用户画像等，该类系统一般使

用前端技术结合可视化组件开发，要求工程师精通 JavaScript、Ajax/JQuery、HTML、CSS 等 Web 前端技术，以及数据可视化技能和工具，例如 D3、Echarts、HighCharts、Tableau 等。熟悉各主流浏览器（IE/Chrome/Firefox/Safari）兼容性问题解决方案和 Oracle、MySQL、MongoDB、Hive、HBase 等数据库查询能力，另外还需了解各种调试、抓包工具如 HTML 类、CSS 类、Debug 类等。

数据应用类系统包括：互联网广告精准投放系统（DSP）、精准营销系统、征信/风控系统、个性化推荐系统、大数据管理平台（DMP）等。该类系统除了会使用前端技术和可视化组件外，还需要结合大数据分布式算法、高并发查询、负载均衡等技术，更侧重 Redis、Nginx、MQ、Zookeeper、Hadoop 等技术。熟悉 TCP/IP 协议和多线程并发技术，同时也要兼具可视化系统开发所应用到的 Web 前端技术、数据可视化技术、浏览器兼容等。

7. 算法开发工程师

算法开发工程师之前一直是一个比较“冷门”且“高深”的岗位，随着大数据应用越来越广泛，使得算法模型在企业大数据应用中越来越广泛。由于每个行业的特性不同，数据模型在跨行业应用时可复用度不高。比如，金融行业应用数据模型进行金融产品的风险控制和反欺诈，建立并优化风险政策。电商及快消行业则应用数据模型进行用户价值评分、偏好预测、商品关联销售和个性化精准推荐。但不管是什么行业，用到的算法是相通的（例如逻辑回归、SVM、神经网络、决策树、贝叶斯等）。

大数据环境下的数据建模开发工程师，除了要求精通传统建模工具 SPSS/Modeler 之外，还要精通 R、Python、Hadoop、MLlib、Mahout 等算法开发组件，了解大数据分析处理（Hadoop、HDFS、MapReduce、HBase、Pig、Hive）等技术内部机制，熟悉 Linux 系统，熟练使用 Shell/Perl/Python 脚本。

2.2.4 产品设计类

项目产品类岗位通常是每个公司不可或缺的岗位，这些岗位是有计划开发数据工作的基本前提，通常决定了一个产品或项目未来的方向和具体实施的概念定义。而项目产品类泛指数据项目工作的前端职位，含项目经理、产品经理、UI、UE 等。

1. 项目经理

项目经理的职能核心是项目宏观管理者和协调者，也是项目实际的总策划人和负责人。

项目经理主要侧重于项目规划、管理、协调工作，重点关注项目进度、质量、成本，通过管理控制项目风险并保证相关成果。跨职能部门进行定期沟通，确保公司内部信息和资源对称；协调项目资源，保证项目正常推进。通过制定实施方法论和项目管理规范来进行整体项目把控，某些场景下的项目经理还会参与需求调研，引导客户需求，编写项目需求文档和相应的技术规范文档等细致工作。对实施完成的项目进行总结，并提供产品研发、项目管理建议。

不同行业的项目经理要求具有特定的从业背景和经验,对于项目开发过程中涉及的管理方法、技术框架、操作规范等都有不同的要求。但较好的号召力、领导力、沟通能力、应变能力和管理能力是胜任该职位的基本前提。

2. 产品经理

产品经理是微观层面落实具体项目需求的关键推动者,也是辅助项目经理进行项目把控的关键,但从职能角度来看通常不具有实权,因此属于典型的“有义务、无权利”的职位。

产品经理主要负责配合项目经理完成项目规划、管理、协调以及规范和文档制定工作,并负责数据相关项目内产品的规划与设计,制定产品开发、设计、跟踪和优化方案。在项目开展过程中需要保持与视觉设计、前端架构、前端开发等部门的沟通并保证产品需求的可理解、可实现、可执行性。根据公司规划,设计产品设计文档、原型设计文档和产品交互原型设计,含界面、流程、功能、组件等。对于整体产品项目质量管理和进度管理,保证项目按照进度完成策划、开发、测试和上线。

由于产品经理需要面对策划、设计、开发、测试、上线的所有环节,因此较强的责任感、创新的工作精神、严谨的工作态度、较强的沟通能力和逻辑判断能力是一个成功产品经理的必备素质。产品项目和生命周期管理的常用工具如思维导图工具、产品原型工具、产品流程工具、版本管理工具、项目管理工具等的熟练应用是必备职业技能。

3. UI

UI也称视觉设计师,主要工作侧重于视觉效果设计,产品视觉效果的好坏主要取决于UI的审美水平和输出能力。

UI的主要工作职责是把握视觉设计趋势,分析产品特点,确定产品整体设计思路和风格;产品、网站、APP等具体产品形态的视觉形态策划;产品、页面、功能、图标等视觉元素的可视化设计,与产品经理、前端工程师共同把握移动产品的用户体验。某些公司的UI可能还会负责广告、营销和包装等宣传物料的设计。

良好的想象力、较高的审美层次和色彩把握能力,熟练使用视觉设计工具如Photoshop、Illustrator、CorelDRAW是必备技能。

4. UE

UE也称交互设计师,主要工作侧重于交互效果设计。很多公司将UI和UE合并到一个岗位职责,总体负责产品的平面和交互设计效果。

UE主要负责维护和更新界面设计标准和规范,负责标准和规范的实施;产品、网站、APP的交互设计工作,分析产品特性和用户的操作习惯和偏好,并设计交互流程、内容及界面;根据需求和用户研究结果,完成界面交互行为和功能的改良,提高网站的易用性;对现有产品的可用测试和评估提出改进方案,持续优化产品用户体验。

在技能要求上,除UI中对于素质和必备技能的要求外,UE还需要了解设计主体(产品等)的商业逻辑、交互工程中的功能需求及信息因素关系,这对该岗位职责的要求更高。

2.2.5 数据挖掘类

数据挖掘类岗位通常是一系列岗位的统称，因为不同公司对于该职位的定义和内涵界定不同。而数据挖掘由于是侧重于应用的岗位，因此通常是围绕某一业务或技术主体进行定义，例如会员数据挖掘、销售数据挖掘、营销数据挖掘等。

数据挖掘类岗位与算法开发类的岗位差异通常是模糊的，一方面由于数据挖掘需要特定的数据统计学、技术开发等特定技能要求，这与算法开发重合；另一方面数据挖掘由于既可以侧重于算法挖掘和应用，又可以侧重于典型场景的业务应用，因此也很难具体固化到某一种岗位角色。通常，该角色可定义为技术类岗位，也可以定义为业务类岗位，但前者居多。

整体来看，数据挖掘类岗位的主要职能包括以下几个方面：

- 负责完善数据挖掘工作体系，优化现有数据挖掘业务落地；
- 负责完善数据挖掘流程、操作规范、标准和监督计划；
- 深入研究业内领先的技术思路，输出具有创新价值的预研项目可行性分析报告及相关实验数据；
- 负责营销（流量）、会员、产品、销售、客服、供应链等公司数据的海量挖掘，并建立、维护和调优常用应用场景如恶意流量预警、库存预测、会员活跃度、会员流失模型等；
- 负责相关数据挖掘项目需求收集、项目制度建立、项目设计开发和结果输出质量把控，通过数据挖掘结果驱动业务执行；
- 配合技术进行数据挖掘模型开发和模型封装，例如决策规则模型、预警模型、流失模型、效果标杆模型、客户生命周期等的建立和维护；
- 负责大数据下传统机器学习算法的并行化实现及应用，并提出改进方法及思路；
- 参与公司大数据架构，负责 BI 实施中的数据挖掘模块算法研究、模型建立和优化，帮助实现数据挖掘和分析平台的建设等。

数据挖掘类岗位对于职能的要求较高，除了具备统计、信息技术、数学等专业学历外，熟悉主流数据库，例如 MySQL、Oracle、SQLServer、DB2 等传统结构化数据仓库以及 NoSQL 等非结构化数据库；熟悉常用的聚类、分类、回归、关联、时间序列等监督式和非监督式算法；熟练使用 SPSS Statics（2009 年之后称为 PASW Statistics）、Clementine（12.0 版本之后称为 SPSS Modeler 或 PASW Modeler）、SAS、R、Python、MLlib 等数据挖掘工具中的至少 1 种，有数据建模经验是从业必备技能。



从某种意义上看，数据挖掘类岗位是数据真正从“数据”到“知识”再到“应用”的枢纽，因此是大数据岗位中的核心职位之一。但是，这种岗位通常只对于拥有“大数据”的公司才有意义，因为只有大数据才有“挖掘”的必要，而小公司由于体量小而导致数据量小而不具备挖掘的必要条件。

2.2.6 数据分析类

数据分析类岗位包含各种各样的非“技术”类岗位，例如战略分析师、数据分析师、网站分析师、用户研究员、商业智能分析员等，这些岗位通常都具有特定的分析应用场景，因此大多数以应用场景来定岗定责。

1. 战略分析师

战略分析师在很多公司也被称为市场分析师，这是一个“高大上”的岗位，它的核心是提高对行业和竞争对手的认知，增加对公司决策层的战略支持。

战略分析师的具体职能通常是根据公司的战略方向，辅助公司决策层（通常是 O-Level）制定中长期发展规划；根据公司规划，协助各中心、各子部门制定战略研究规划并进行课题跟踪和持续输出；收集行业重要信息，包括重要盈利模式、重大技术革新、新技术发展趋势、市场格局重大变化等；过滤公关信息，建立竞争对手档案库，全面把控竞争对手动态；跟踪、分析、研究行业发展情况，捕捉行业发展新机会，为集团的战略决策提供依据。

战略分析师对于从业者的个人素质要求极高，它要求对业务生态和体系具有相当丰富的工作经验，具备敏锐的市场和行业洞察及快速的业务理解和学习能力，以及从宏观角度总结、分析和归纳问题及方法的能力等。另外，对于常见的市场和战略分析框架的熟练应用、敏锐的市场嗅觉、较强的逻辑思维和沟通能力是必备的个人素质。

2. 数据分析师

数据分析师是一类职位的统称，通常数据分析师会定位于解决某一类问题而带有业务主体特征，例如营销分析师、会员分析师、运营分析师、商品分析师等。但无论如何定位，其基本工作职能如下：

- ❑ 搭建公司数据分析体系并负责日常数据质量、报告、结论的把关；
- ❑ 建立业务主体档案库，并通过效果预测模型，辅助业务主体计划和 KPI 的制定；
- ❑ 完善业务主体的画像，并通过多种价值模型做业务主体分群、分类；
- ❑ 识别业务主体中的虚假、异常、流失等信息，建立相应的预警系统；
- ❑ 业务主体活动效果评估分析，并通过多种数据结果提升目标转化率；
- ❑ 建立业务主体效果标杆，提高业务要素的利用率并建立最优化效果评估模型和组合应用模型；
- ❑ 协调利益相关者对如何使用研究和分析结论的想法，以支持业务计划和战略排序；
- ❑ 针对特定场景建立生命周期模型，并针对不同场景和阶段下的实际情况建立相应的分析思路和方法，辅助于业务主体优化；
- ❑ 根据业务和公司需求，跟进专项分析项目进度，撰写日常和专项报告并优化业务落地动作等。

该岗位要求具有一定的统计学、数学、计算机科学等专业背景，同时了解数据分析的基本概念和常用方法，熟悉常用业务主体中的指标及应用场景，具备较强的逻辑分析能力和报

告书写、业务沟通能力。对于常见的数据库取数工具如 SQL、数据库客户端以及数据建模和挖掘工具、Excel 和 SPSS 等统计和分析工具也有一定要求。

3. 网站分析师

网站分析师是数据分析类中一个较为特殊的职位分类，从工作形态看，除服务器需要 IT 部门配合进行相应配置、调试和部署外，网站分析师几乎可以独立完成从数据采集、存储、计算、分析到数据应用的完整流程。网站分析师的工作主体和对象是以网站为主体的业务主体，包括营销部门、网站运营部门、用户体验部门、前端产品部门等。

网站分析师的主要工作职责如下：

- ❑ 根据业务需求进行网站检测代码方案的制订、实施和后期维护；
- ❑ 监控网站日常数据，为公司各级部门提供需求数据、日常报告；
- ❑ 根据业务和公司需求，撰写专项分析报告，提供专业决策支持；
- ❑ 对网站流量、运营数据进行跟踪和分析，尤其是对站外投放渠道、站内运营效果进行深入挖掘；
- ❑ 对用户数据进行深入分析，如页面点击分布、用户行为习惯等，了解用户需求并提出优化改善建议；
- ❑ 网站流量系统管理、维护，跨部门沟通协作与项目推进等。

由于从事网站分析工作的特殊性，往往需要熟悉网站分析系统部署规则、代码和语法，同时熟练应用不同的网站分析工具如 Google Analytics（简称 GA）、Omniture（现在名为 Sitecatalyst，是 Adobe Analytics 的核心）、Webtrends、Webtrekk 等，对于网站分析和数据分析的基本概念和方法以及常用指标及应用场景也要熟稔于心。除此之外，还要具备特定行业的从业经验，特定的专业背景如营销、计算机等是其加分项。

4. 用户研究员

用户研究员是公司中针对用户研究的岗位，主要目的是通过不同的研究方法来提升用户满意度、降低用户流失并提升用户生命周期价值等。

用户研究员的主要职能是组织各种用户研究项目，与产品经理交互和研发团队沟通，发现用户体验提升的工作点；独立完成用户研究项目的全套流程，包括需求分析、方法设计、数据分析、结论提取以及报告撰写；建立特定的用户研究项目，包括用户群体的行为分析、目标用户验证、产品体验验证、可用性测试、满意度研究等；建立和推动产品以用户为中心的工作制度和流程优化。

由于用户研究的主体是用户，因此需要该岗位的人员具有人机交互、心理学、社会学或相关专业背景，熟悉不同的研究方法和流程，对于研究数据具备一定的统计和数据分析能力及提取结论的能力，较强的沟通和业务理解能力、敏锐的洞察力和快速学习能力是岗位的加分项目。

5. 商业智能分析员

商业智能分析员也叫 BI 分析师，是借助或依托于 BI 系统进行数据分析的岗位。该岗位

通常是在企业内部已经建立起 BI 体系并搭建 BI 系统的前提下产生。

该岗位的主要职责是通过 BI 进行日常数据处理、监控和统计分析并支持运营活动，参与制作时间分析报告并为决策层提供数据支持；参与 BI 系统的搭建、优化和开发，进行或协调测试，以确保情报的定义与需求相一致；根据业务需求配置相关的 BI 模型和报表并为业务主体使用；BI 系统的日常管理和维护，包括维护或更新的商业智能工具、数据库、仪表板、系统或方法等。


商业智能分析员除了需要具备数据分析师的有关数据分析基本经验和能力外，还需要熟练掌握 BI 系统的部署、实施、配置、规则和应用知识，能通过 BI 工具满足不同的应用场景。

2.3 大数据制度和流程规范

2.3.1 制度和流程规范意义

规范化管理是企业中一项艰巨的且需要持续改进的工作，它是企业各项工作正常有效开展的基础，是企业健康有序发展的有力保障。大数据制度和流程规范作为企业规范化管理的一部分，对于大数据工作的开展至关重要。大数据制度和流程规范建设的意义主要侧重于三个方面：

- 可以保障企业内部大数据系统和周边业务系统运作的有序化、规范化、流程化和标准化，可以降低沟通成本并提高工作效率，保证最终的工作产出。
- 可以通过制度性措施界定各事业群、事业部、体系、中心、部门间的利益主体和权责范围，是有机开展工作，避免推脱、不作为、越权工作的重要途径。
- 通过制度性约束可以降低业务运作风险以及数据安全风险，这是企业开展大数据工作的基本前提。

 **注意** 通俗而言，制度和流程规范不是必须的，或者说不是所有企业都需要严格的制度和流程规范。在实际应用中，制度和流程规范通常适用于大中型企业，为了提高企业运转效率而采取建立现代企业制度的方式；而对于小企业而言，灵活的管理方式、直接高效的沟通机制和更扁平化的直接管理可能更适合真实运营的需要。因此，这里的制度和流程规范的试用对象更多的是针对大中型企业。

2.3.2 制度和流程规范内容

制度和流程规范类内容大致可以划分为两类：

- 工作制度，这类文档对其范围内的人员进行约束，常有“制度”“规范”“规章”等字眼出现，这类内容不能被随意修改；
- 工作模板，这类文档是相应人员开展工作的参考内容，可供其直接应用于大数据工作开展，也可根据实际情况进行修改。

大数据制度和流程规范建设涉及大数据工作中的所有环节，从大数据的工作体系看，包含以下几个部分：

1. 基础平台类

基础平台类规范提供服务器测试和正式环境的系统运营、服务维护、应用维护管理的范围、目的、性质和原则，通常以系统运维管理规范或制度的形式存在。该规范适用于开展系统运维活动涉及的各类组织及其落地操作的工程师。

规范主要涉及的内容包括：

- ❑ 机房环境，包括安全系统、空调、UPS、备用发电机、供水、供气、排污等；
- ❑ 基础服务器，包括主机系统、存储/备份系统、终端系统等；
- ❑ 网络设施，包括交换机、网络、通信、电缆等；
- ❑ 应用系统，包括内部办公系统、门户网站等应用系统；
- ❑ 业务系统，包括内部开发以及外部购买的业务系统等；
- ❑ 中间件，包括配置信息、故障信息、性能信息监控等；
- ❑ 供应商系统，包括基础设施和应用系统的供应商以及 IT 运维服务的供应商系统；
- ❑ 云服务系统，包括采购云端的固定投入平台以及按需付费的弹性平台系统等。

除了基本运维信息涉及的系统软硬件运维管理外，还可能包括权限管理、数据管理、系统监控、系统培训等内容。

基础平台类规范的主要核心是通过各种标准化和流程化规范保证系统的可用性和稳定性，规范中需要兼顾到不同角色的负责人和职能分工、量化的工作标准和响应时间、操作流程和方法、问题沟通工具和流程等。除上述规范性工作流程外，建立针对突发事件应急预案和防护策略也是规范的重要组成和安全的应急保障。



注意 大多数企业中除运维工程师自己发现并解决问题外，其他系统或部门人员也会反映相应的问题，此时通常会通过一个名为“IT 工作台”或“IT 服务台”的角色对涉及的大数据相关事务进行统一收集、分配、处理和反馈管理。

2. 数据管理类

除 CDO（首席数据官）外，数据管理类的主要操作或管理对象是数据，因此本小节主要讨论的内容是有关数据及其数据周边的制度及流程规范。数据管理类规范的主要存在方式为数据库管理规范以及相应的流程规范，它主要针对数据进行管理，降低数据被非法生成、变更、泄露、丢失及破坏的风险。该规范适用于 DBA、数据库管理工程师、数据安全管控师等。

规范主要涉及的内容包括：

- ❑ 数据范围，涉及所有的业务系统、职能系统和 IT 系统数据；
- ❑ 数据环境，包括所有的测试环境和生产环境数据；

- ❑ 数据公司, 适用所有集团、总部、子公司和分部等各类数据相关组织;
- ❑ 数据有效期, 大多数数据都是有有效期的, 不同有效期状态下的数据应该有针对性的管理策略、存储介质;
- ❑ 数据安全规范, 包括数据安全定义、接触、接入、备份、同步、授权、认证、加密、操作日志记录等;
- ❑ 数据操作规范, 包括数据的新增、修改、更新、删除等数据变更规范, 数据加密、解密等脱敏和安全规范以及数据提取、分发、打印、登记等流通规范;
- ❑ 数据库管理规范, 包括用户角色管理、数据库管理、系统升级维护、数据库安全管理等。

数据管理类规范是数据安全的必要保障, 也是开展所有数据工作的基本前提, 因此是每个公司必须具备的一类规范和流程制度。出于数据安全第一的考虑, 必要的数据流程和权限申请管理是必不可少的。



注意 大多数企业的数据操作都是针对非生产数据进行的, 生产数据都是作为原始数据进行保存, 然后将原始数据同步到附属库或从库的库表中进行操作。保存至少一份原始数据是保证数据在任何时间都处于高可用状态的前提。

3. 技术研发类

技术研发类规范主要用于在团队协作开发的情况下, 保证架构、编码、测试等各个环节的一致性、可读性、可重用性、程序健壮性、可移植性、可维护性。该规范是提高团队协作开发效率和软件质量的必要保障, 也是降低后期维护成本的重要举措。

技术研发类规范从流程上可分为两大类:

(1) 文档规范

技术研发过程中, 需要根据不同的项目撰写相应的研发文档, 包括概要设计文档、详细开发文档、质量校验文档、集成测试文档等, 这些文档是日后进行技术研发的基础。文档需要详细记录产品的研发背景、蓝图、目的、原则、阶段、里程碑、排期、内容、约束和前置条件、沟通计划、机会风险等, 其阅读对象是项目成员以及相关的研发工程师。该类文档是项目执行的参考, 为项目按时交付、项目测试、质量跟踪以及后续开发等提供了书面依据。除了面向技术研发的文档规范外, 还有一类面向客户的文档规范, 这些信息会在“项目产品类”规范中具体介绍。

(2) 代码规范

代码规范是面向技术研发人员在产品或系统开发时具体实施的操作性规范, 它涉及开发过程中撰写代码时的各个方面。规范主要涉及的内容包括:

- ❑ 文件结构, 包括头文件、定义文件、其他文件的路径、目录、结构等具体定义;
- ❑ 程序风格, 包括空行、空格、缩进、续行等定义, 这是通过逻辑关联分组、组之间的关系, 提高可读性的保障;

- 命名规范，比较著名的命名规则当推“匈牙利”命名法，该命名规则的主要思想是“在变量和函数名中加入前缀以增进人们对程序的理解”。命名规范中包含了对库、包、类、域、方法和声明的具体定义；
- 注释规范，包括文本注释、块注释和单行注释的注释内容、方式、位置等约束，对于文件头和函数头的注释内容包括功能、参数、返回值、设计思想、调用函数、日期、修改记录、设计者信息；
- 类、函数和方法，包括对象本身的参数和返回值，对象相关的声明格式、可选元素、类体成员、类内成员顺序、方法释义、影射关系、引用等；
- 错误处理：对于可能出现的错误信息的提示方法、处理过程和逻辑的定义；
- 兼容性规范：对于程序开发过程中涉及同一程序或语言由于版本不同可能导致的兼容性或功能问题，以及适配周边系统环境的兼容性问题的处理；
- 资源调用：区分 Debug 版本和 Release 版本，同时对系统软硬件资源进行配置，例如指针、资源释放等。



注意 在项目建立之初，通常所有的文档规范就需要制定好，这些规范或材料通常会通过知识中心或知识库作统一管理，这些知识库或知识中心可以集成到 SVN、Bug 管理工具、Wiki 工具、知识管理系统以及其他项目管理工具或公司系统中，以便于知识和制度共享以及信息发布。

4. 项目产品类

项目产品类的规范和制度主要针对项目实施和产品实施的整个项目制定的相关规范。项目产品类的规范和文档的主要对象是项目中不同阶段的参与人员，包括项目、产品、设计、开发、运维等人员。

项目产品类规范和制度涉及每个文档生命周期的始末，从创建、审批、发布、变更、分发、追缴、归档、废止到恢复等。

常见的项目文档通常分为 4 个阶段分别进行定义：

(1) 立项前的市场分析类

立项前的市场分析类文档通常包括市场调研报告、可行性报告、风险评估报告等。这三份报告都是针对市场调查、收集、整理和分析后，结合市场规模、特点、容量等对项目的可行性、前景、利弊、机会进行分析，常用的维度包括宏观环境、竞争对手、自身情况、目标客户等，分析模型包括 SWOT、PEST、STP、4P、4C、波士顿矩阵、五力模型、生命周期模型等，分析方法包括系统分析法、结构分析法、演绎分析法、定量与定性分析法、案例分析法、复合分析法等。

(2) 立项后的规划分析类

立项后的规划分析类主要指的是在项目立项后，为了整体项目的开展而进行的整体规划

和分析工作,通常产出物为项目开发计划文档。项目开发计划中通常涉及对项目前景、主要内容、参与范围和人员、人员角色定位与分工、计划实施分解和进度跟踪、关键里程碑及产出交付物、前置和约束条件、预期和最晚交付时间、验收标准和评审、成本和预算评估、风险评估和控制等。制订开发计划需要不断细化和丰富,开发计划是项目经理管理和跟踪的依据,可起到指导项目组的整体进度调控和日常工作跟踪的作用。当实际开发情况与开发计划偏离较大时,应修正开发计划或实际开发情况。

(3) 实施中的开发规范类

项目开发实施过程中,在不同阶段涉及不同的文档和规范,从实施的阶段来划分可分为产品类文档、技术研发类文档、测试类文档三类。

□ 产品类文档包括软件/产品需求说明书、UI/UE 设计规范、用户交互设计规范等。

□ 技术研发类文档在 2.3.2 节中有具体解释,在此不再赘述。

□ 测试类文档包括测试计划书、测试评估报告、问题追踪报告等。

(4) 实施后的验收类

项目实施完成,通常需要交付一系列文档,可能包括软件/产品验收报告、项目总结报告、运营管理手册、软件质量保证计划书、用户操作手册、帮助文档和 FAQ 等。

除此以外,项目进行过程中,会贯穿着多种项目跟踪类报告,包括开发进度月报、阶段性总结报告等,这些报告根据实际排期和里程碑计划情况安排即可。



注意 对于项目文档的管理,可以使用 SVN,但通常更多的是使用专门的项目文档管理系统,例如 VSS、HFS、TeamOffice、SharePoint 等。但采用何种工具,具体根据企业需求和实际情况进行选择即可,适合的才是最好的。

5. 数据挖掘、分析和应用类

数据挖掘、分析和应用类规范是针对开展数据工作中,涉及非技术开发类的数据挖掘、分析和应用类的流程和方法而制定的规范,其目的是保证数据工作的及时性、有效性,以及结果的正确性和可应用性。

按照数据工作的项目流程,通常分为需求沟通、需求提报、商业理解、数据准备、数据挖掘(含分析)、部署实施 6 个阶段,如图 2-6 所示。整个过程应该通过一定的工具和流程规范进行控制和集中管理,否则数据工作就会失控并且毫无落地价值可言。

(1) 需求沟通

需求沟通已经在数据需求管理中提到,不合理或不可行的需求将被直接驳回。正常情况下,需求沟通当天应该反馈沟通结果。对于需求中由于主客观原因无法实现的、错误的需求,无法落地的需求以及重复需求应该予以驳回。在这个过程中,建议采用数据对接人制度,将不同业务部门负责数据对接工作的人员固定下来。

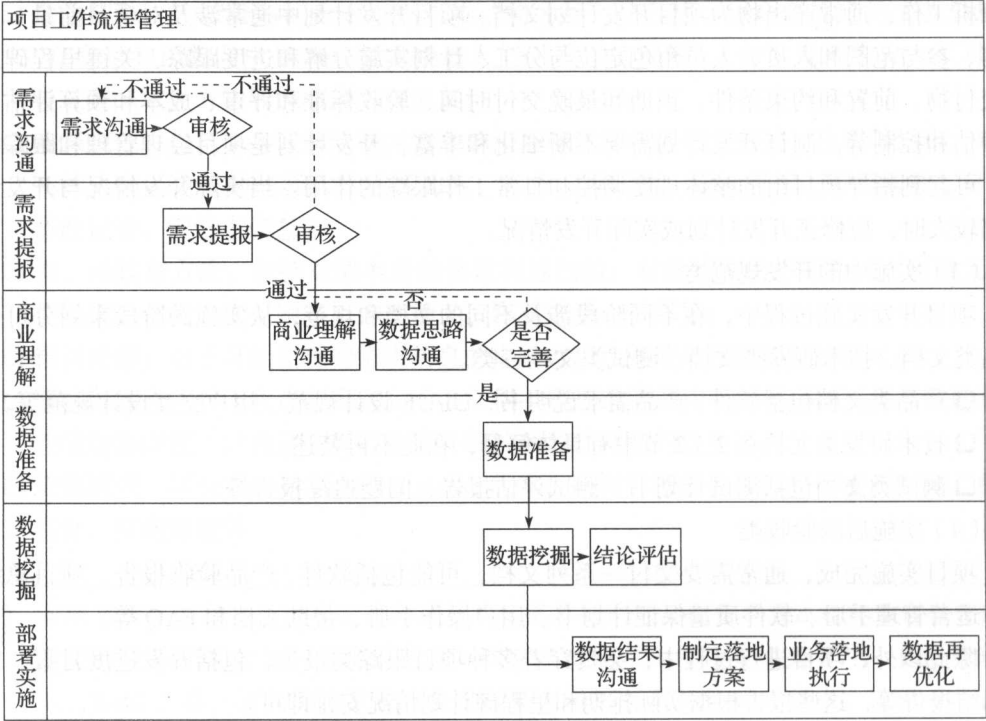


图 2-6 数据项目工作流程

注意 很多时候业务需求不能落地，例如数据提取工作只是为了验证工作效果，对于此类简单的需求需要通过培训、开放权限等方法让业务自行实现。数据部门不应该把时间浪费在这种价值太低的工作上。

(2) 需求提报

在需求提报阶段，不符合公司利益或可能对公司产品产生负面影响的需求也将被驳回。需求提报和审批根据不同企业的流程复杂程度和实际审批效率而定，通常在 1 ~ 7 天之内完成。当续期需求中涉及公司敏感性指标、较高的数据权限、加密和解密处理、外部数据处理请求等特殊内容时，通常需要通过公司内部 OA 类系统进行申报和审批。

提示 数据需求提报管理是数据需求审核中不可或缺的步骤，在很多大型企业中往往是企业级流程管理的重要部分。需求提报管理过程中，企业领导层从企业全局的角度把控数据需求是否合理，其决策关乎整个公司而非数据部门。

(3) 商业理解

商业理解是将业务语言转化为数据语言的过程，目的是确定业务预期效果的维度、范围

等,这个阶段通常需要2~3天的工作时间。商业理解阶段包括两部分内容:

- 商业理解沟通:数据部门理解业务部门具体需求的过程。
- 数据思路沟通:数据部门将业务理解转化为数据分析和挖掘思路的过程。

本阶段的产出是数据分析和挖掘工作思路,通常以思维导图的形式输入并加以沟通确认。如图2-7所示为渠道画像分析思路。

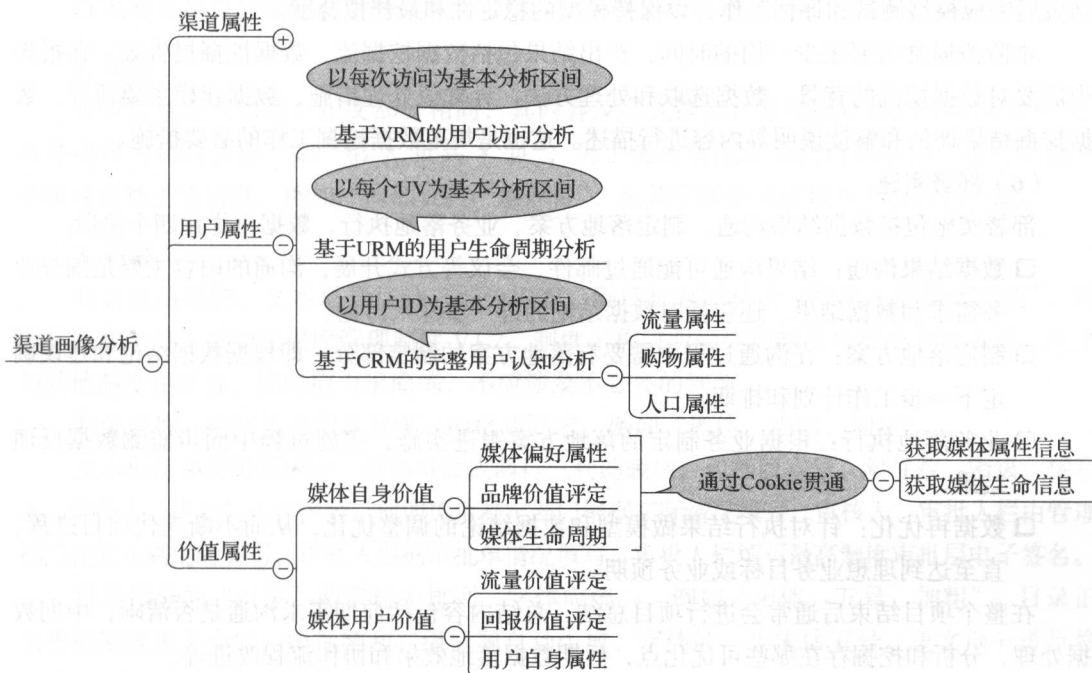


图 2-7 渠道画像分析思路

(4) 数据准备

数据准备是对即将进行的分析和挖掘工作进行预处理,包括从数据仓库中取数、验证数据质量、数据特征提取、异常值处理、数据转换和合并等,为后期的数据分析挖掘做准备。这个阶段是费时但非常重要的工作,前期这个工作做不好会直接影响数据质量,从而影响结果的可信度及稳定程度。

该项工作通常需要1~4天的工作时间,根据原始数据质量及数据量级的不同而有所差异。阶段性数据产出结果为数据质量报告以及清洗之后的数据。



提示 数据准备是数据工作中的难点,很多时候由于原始数据质量较差或数据从业者自身工作经验和能力不足,导致大量时间耗费在数据准备和清洗阶段,使得后期数据价值挖掘的投入精力不足,从而影响数据结果和价值产出。因此,这个阶段一定要在保证数据质量的基础上缩减投入时间。

（5）数据挖掘（含分析）

经过前期的各项准备工作，接下来就开始了数据工作的核心环节——专项分析和挖掘工作，包括常用的描述性数据统计方法，LDA、PCA 等数据预处理和转换方法，时间序列、分类、聚类、回归、关联和序列关联、规则提取等传统数据挖掘和建模方法，以及协同过滤、神经网络、深度学习、自然语言处理等监督式和非监督式学习算法等，并在专项分析或建模结束后完成模型测试和评估工作，以保持模型的稳定性和最佳拟合度。

本阶段通常需要至少一周的时间，产出结果包括数据挖掘流、数据挖掘报告等。在报告中需要对数据挖掘的背景、数据选取和处理方法、异常值处理措施、数据建模主要流程、数据挖掘结果评估和解读说明等内容进行描述。这也是规范数据挖掘工作的必要措施。

（6）部署实施

部署实施包括数据结果沟通、制定落地方案、业务落地执行、数据再优化四个阶段。

- 数据结果沟通：结果沟通可能通过邮件、会议等方式开展，沟通的内容主要是围绕业务需求和数据结果，还包括对数据结论的进一步深入讨论。
- 制定落地方案：在沟通过程中需要有落地方案的制定部分，即根据数据结论和建议确定下一步工作计划和排期。
- 业务落地执行：根据业务制定的落地方案跟进实施，实施过程中同步监测数据反馈结果。
- 数据再优化：针对执行结果做模型和数据结论的调整优化，从而不断迭代项目进程，直至达到理想业务目标或业务预期。

在整个项目结束后通常会进行项目总结，总结内容包括前期需求沟通是否清晰，中期数据处理、分析和挖掘存在哪些可优化点，后期数据落地效果和协作流程改进等。



注意 不是所有的项目都以成功结束，很多时候由于主客观原因导致项目失败。但项目失败也是一种知识成长的过程，此时更应该与业务部门一起深入总结，以避免日后出现类似的失败问题。

本阶段的时间大概为 2 周左右，具体以业务落地执行时间为主。产出结果包括业务落地计划方案、落地执行结果评估报告等。

2.3.3 制度和流程规范模板

由于不同的制度具有不同的内容指向性，因此不同类型的文档规范的内容主题不同。对于不同类型的规范和制度，通常规范会涉及以下几个方面：

1. 页眉

页眉信息英语封面（如果有）应与正文部分相同，由公司名称或 Logo、制度编号、制度

名称及发布日期组成。制作页眉时需要注意以下几点：

- 文字格式：统一使用一种格式，通常使用宋体五号字。
- 制度编号：由中心简称、部门简称和分类顺序号三部分组成，由相应的管理部门在制度发布时统一编制并添加。
- 制度名称与封面页顶端所注标题一致。
- 发布日期以公司制度审批最高层签发日期为准，由管理部门在制度发布时标注。

2. 页脚

页脚信息应与封面、正文部分相同，其内容及形式固定，制度起草部门不应擅自修改，具体内容为页码信息，如“第 × 页共 × 页”，为了提高规范或文档的保密性，还可增加一些版权或禁止类信息，例如“内部资料严禁外传”，且其字体格式应该与页眉保持一致。

3. 封面

封面包括标题、文本框及目录三部分内容。制度标题结构为“管理主题管理制度”，后面标注版本号，例如数据库管理制度 V1.2。制度名称应明确体现制度规范的主要事项，使之与其他制度相区分，同时应力求简练，不应涉及不必要的细节。

制度名称一栏字体必须具有统一的格式要求，例如“黑体，小四，加粗”。

文本框行和列需要固定，例如可做成两行三列的表格，包括版本号、附件数、密级、撰写人、审核人、审批人六项内容。前四项由制定部门根据实际情况填写，审核人、审批人栏由管理部门在发布制度时填写，审核人根据审批单情况填写，审批人栏填写最高制度审批层电子签名。

目录部分的“目录”两字居中排列，字体应统一，例如“宋体，五号，加粗”；目录正文根据制度正文中的一级标题和二级标题自动生成，字体统一为宋体五号。正文部分进行修改后，应同时更新相应目录。



注意 通过 Word 中的引用功能来生成目录是一个维护目录和内容一致性的有效方法。

4. 正文

制度正文包含目的、范围、名词解释、职责、管理制度、工作流程、注意事项、附件八部分内容。

- 目的（必备）：简要说明制度出台所需要解决的问题，或要达到的目标，以及制度的作用和意义。
- 范围（必备）：说明制度的适用范围（适用于哪些部门、人员、事项及工作环节）和发布范围（制度需要在哪些部门、区域或人员范围内发布）。
- 名词解释（可选）：主要对制度中出现的专有名词进行解释或界定范围，以便大家准确理解。一般包括需要相关知识、技术背景或工作经历的人才能理解的专业术语，以及使用中有不唯一确定含义的词语和其他特定含义的用语。

- 职责（必备）：主要确定制度中各事项、环节的实施主体部门、岗位，以及与此相关的其他部门、岗位的分工、各自的权限和相互间的协调关系。
- 管理制度（必备）：管理内容和管理方式是管理制度的主体内容。管理内容与管理要求主要规定该制度管理的业务内容、工作标准及具体要求、信息反馈的渠道、时间等；管理方式主要规定对管理事项执行检查、考核的负责部门、内容、程序、时间、方法等。
- 工作流程（必备）：对制度涉及的特定业务工作的流程予以详细描述，必要时可以辅以流程图示意图或标准流程图。
- 注意事项（可选）：对制度理解与贯彻执行中的需要予以特别强调、提起重视或有特殊要求、须格外注意的问题做出说明，例如制度的生效或实施日期、制度的执行部门、解释权限等。
- 附件（可选）：附件即为制度内容中所要求的相应记录及管理表单（空白模版），须为公司其他管理制度文件中所未包含的。制度内容中首次引用附件时，需在该附件名称后作“（见附件×）”标注。

正文部分各部分序号使用多级列表形式，一级列表顶格排列，以下一般依次缩进2字节。

正文中一级标题一般设置为“标题1”样式，制度的主要、重点部分的二级标题可以设置为“标题2”样式，以便在目录中引用。正文部分字体需统一（例如统一使用宋体五号字，标题加粗，段落设置段前段后均为0，行间距一律为1.4倍）。

5. 附件

附件应按文中所列顺序置于正文之后，一般情况下各附件独立排列。对于管理制度的附件通常包括管理汇总信息表和新增管理内容表两部分。

6. 附录

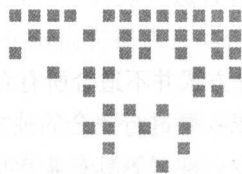
其他需要体现在制度中的特定内容或指导信息。

2.4 本章小结

职能规划是企业大数据落地的基本前提之一，本章从大数据组织架构体系、职位构建体系和制度流程规范三个方面介绍了如何进行职能规划。在规划并建立好职能体系之后，企业大数据工作便进入正式落地规划阶段。

本章需要读者重点掌握的知识点包括：

- 掌握常见的大数据职能架构划分方法并根据企业实际情况选型；
- 掌握大数据不同职能中心/部门的职能以及岗位划分方法；
- 掌握不同职能内部需要建立的制度和流程规范，进而保证数据工作的规范性开展。



企业大数据解决方案

企业在开展大数据战略时，往往面临多种实施解决方案的路径可供选择。由于企业需求、背景和环境的不同，所适用的解决方案和实现方式也会有所差异。对大多数企业而言，适合企业的解决方案才是最好的，而非是那些价格最贵、功能最全的。

不同解决方案之间有哪些区别？它们各自对于企业的要求和限制条件是什么？企业应该如何根据自身情况选择最合适的大数据解决方案？本章将重点介绍这些问题。

3.1 企业大数据解决方案实现方式

工欲善其事必先利其器，企业大数据解决方案既包括大数据产品和工具层面，又包括服务层面。企业大数据解决方案根据实现方式的不同可分为完全独立研发、直接购买第三方解决方案和借助第三方的力量进行联合开发三种。

3.1.1 独立研发

独立研发指的是企业内部通过组建大数据中心或部门，独立进行大数据项目的研发。企业独立研发大数据平台，在数据安全、技术可控、后期扩展等方面具有重要意义。

❑ **数据安全：**独立研发过程中的所有数据从输入端到输出端的整个流通都在企业内部进行，数据不会因为平台的开发以及外部人员的介入而产生数据安全性问题。

❑ **技术可控：**大数据平台的所有技术、组件、功能、代码等均由企业完全控制，这对于后期架构重构、代码优化、接口扩展、系统解耦等非常重要。

❑ **后期扩展：**独立研发的大数据方案在应对业务需求更变、数据源增加、数据环境异构、

系统部署升级、硬件平行扩展等运维过程中，会表现出更好的适应性、灵活性和扩展性。

但是，这种方式并不适合所有企业，它对企业来讲具有以下几个方面要求和限制：

- ❑ **技术要求：**要进行完全的独立开发，对于大数据解决方案的所有环节，例如架构、运维、开发、部署等具有非常高的技术要求。
- ❑ **解决问题的能力：**在大数据解决方案的开发和落地过程中，会面临各种挑战与问题。有些问题来自于客观运行环境，也有些来自于技术能力和业务对接过程中。这就要求大数据项目的策划者和推动者具有较高的分析问题、解决问题的能力。
- ❑ **时间投入：**大数据技术的开发往往需要一定的人力和时间投入作为保障。与此同时，外部市场环境的瞬息万变导致大数据项目的价值需求也会与这种时间限定之间产生矛盾，因此企业也需要有效协调二者的关系。
- ❑ **资源精力：**由于大数据解决方案是服务于企业所有体系和部门的，因此在实践过程中需要投入很大的资源和精力进行资源协调和利益平衡等；再加上企业初次实施大数据项目时的经验有限，因此在处理这些问题时需要投入的资源更多。
- ❑ **行业专家：**大数据解决方案不是纯技术性的工作，而是结合了技术、数据和业务的全视角方案，这就在客观上要求企业内部需要有一批了解技术、数据和业务的复合型专家以及各个细分领域的资深带头人，这样才能保证方案落地的可靠性、有效性和价值性。

综上，自主研发的方式更适合具有下列特点的企业：企业内部有一批专家、具有非常强的解决问题的能力、较强的技术实力、充足的资源保障、对大数据没有较强的时间紧迫性要求。除此以外，企业对于数据安全、技术可控、后期运维方面的需求较为明显。

3.1.2 第三方解决方案

由于国外市场的开发性、企业运营的成熟性以及法律法规保障的完善性，很多国外的大型企业尤其是上市公司通常更愿意直接购买成熟的大数据解决方案。直接购买第三方成熟的解决方案具有如下优势：

- ❑ **标准解决方案：**利于在内部各个办事处、子公司、子体系内的推广应用，整个部署、管理和应用都是相对标准化、流程化、规范化的，符合现代企业运营的要求。
- ❑ **弹性付费方式：**第三方解决方案尤其是云服务都允许客户根据自身需求进行资源的弹性配置，然后再做弹性付费，这是一种非常灵活的付费方式。
- ❑ **动态资源配置：**对于企业大多数需求的变更，都可以通过灵活的资源设置来匹配，这是一种简易且高效的资源配置与供需匹配方式。
- ❑ **行业经验积累：**很多第三方解决方案都会根据行业做聚焦和细分，并推出行业性的垂直解决方案，提供比较成熟的环境配套、组件搭配、框架优化和应用模型等，这对于企业快速将大数据进行成果转化具有极其重要的指导意义。从一定程度上看，不同的大数据技术方案在技术本身差异不大的前提下，成熟的行业应用和价值落地模型则是

企业大数据价值差异化的关键。

- ❑ 自动化运维服务：基于云平台的解决方案，服务供应商将提供自动化运维管理能力，这将大幅度降低企业日后的运维成本。
- ❑ 可靠的防护体系：安全一直都是 IT 关心的焦点之一，云平台的解决方案服务提供商都会提供全面的安全解决方案，并通过全方位纵深防御体系来保障云服务的安全，企业无需为安全担忧。

第三方解决方案在提供了一定的安全性、便利性、可靠性的同时，也会带来一定的不足：

- ❑ 无法提供定制化服务：即使能进行弹性配置，前提也是标准化的组件或服务，其中都是将行业内的通用应用规则进行固化，因此无法根据不同企业的需求进行定制开发，这在客观上会限制企业内部个性化需求的实现。
- ❑ 关键技术的不可见：第三方解决方案都会对关键技术、组件等进行封装或加密处理，使得其中的关键技术不可见，这将不利于企业的技术积累和创新。
- ❑ 云服务的可靠性：大多数的云服务在正常情况下都会提供相对稳定的可靠性，但在某些极端条件下，云服务的可靠性会面临巨大考验。比如，在双 11 这样的大型活动中，某云服务商调用所有可用资源来保障其自身的可靠性，进而会对其他客户对云平台服务的可靠性造成威胁。
- ❑ 很难进行二次开发：第三方解决方案由于对关键技术的封装将严重限制企业根据自身需求进行二次开发，即使提供了一定的 API 或 REST 服务，也只能在既有功能下进行二次调用。
- ❑ 云数据的安全性：基于云端的大数据解决方案应用的前提是将数据放到云端（通常是第三方服务平台），这对于企业意味着数据存在安全隐患和泄露风险。对很多大型企业来讲数据即企业机密，尤其是有关企业核心竞争力的数据将不被允许在企业外部流通。

综上，直接采用第三方解决方案更适合希望借助第三方的平台快速进入大数据工作状态，借助其成熟经验将大数据的价值迅速落地，并在后期运维过程中不想投入太多的企业；但对于技术完全可控、二次开发需求大、数据安全要求高的企业将不适用。

第三方大数据方案服务商非常多，甚至可以说有一个大数据生态圈。在这个生态圈中既有能够实现端到端的完整链条的整合解决方案，也有侧重于数据采集、数据预处理、数据存储、数据挖掘分析、数据可视化等环节的垂直型方案。国内的服务提供商包括阿里巴巴、百度、腾讯、华为等，国际大数据巨头包括 Amazon、Oracle、SAP、Dell、TERADATA、EMC、Opower、Splunk、Intel、Google、Microsoft、IBM、HP 等。

以阿里巴巴为例，阿里巴巴提供的阿里云是中国最成熟也是应用最为广泛的大数据解决方案之一。阿里云不仅提供大数据方案的服务，更提供了包括云计算、安全、域名与网站等不同服务。在大数据领域，阿里巴巴提供的服务叫做“大数据（数加）”，其中包括数据应用、数据分析展现、人工智能、大数据基础服务四类。如图 3-1 所示为阿里云大数据解决方案内容。

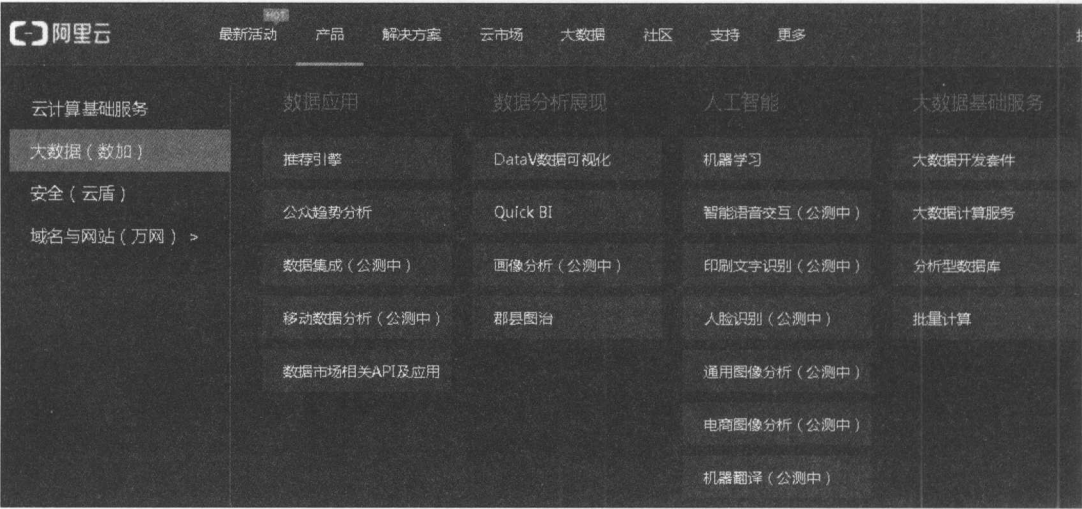


图 3-1 阿里云大数据解决方案

1. 数据应用

推荐引擎

推荐引擎 (Recommendation Engine, RecEng,) 是在阿里云计算环境下建立的一套推荐服务框架, 目标是让广大中小互联网企业能够在这套框架上快速地搭建满足自身业务需求的推荐服务。

它基于阿里云的一体化部署 (SaaS), 为推荐业务定义了一整套规范, 同时提供了默认算法模板以及自定义功能; 支持接入实时日志, 以及实时修正 API; 通过多种测试手段和监控方式为业务决策提供参考。如图 3-2 所示是阿里云的推荐引擎配置界面。

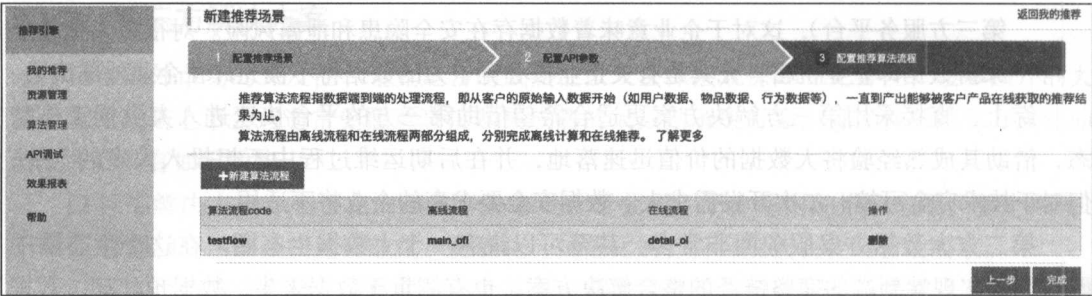


图 3-2 阿里云的推荐引擎

公众趋势分析

公众趋势分析是基于全网公开发布数据, 结合媒体传播路径和受众群体画像, 利用语义分析、情感算法和机器学习等大数据技术, 识别公众对品牌形象、热点事件和公共政策的认知趋势。

它全面覆盖全网公开的数据 (千万源站, 每日更新 20 亿网页), 能最快 2 分钟级别获得数据; 通过机器学习、自然语言、文本处理的协同处理等提供精准有效的结果, 结合分级告警、智能分析、协同处理和深度集成等为客户提供丰富的服务。如图 3-3 所示为阿里云公众趋势分析报表。

移动数据分析

移动数据分析（Mobile Analytics）是阿里云推出的一款移动 App 数据统计分析产品，提供通用的多维度用户行为分析，支持日志自主分析，助力移动开发者实现基于大数据技术的精细化运营、提升产品质量和体验、增强用户黏性。

移动分析能采集用户行为和应用性能数据，通过秒级的实时计算，为客户提供丰富的维度统计报表。同时，它还能通过与移动加速、移动推送、移动域名解析等合力为移动开发者提供更完善的移动服务。如图 3-5 所示为阿里云移动数据分析报告界面。

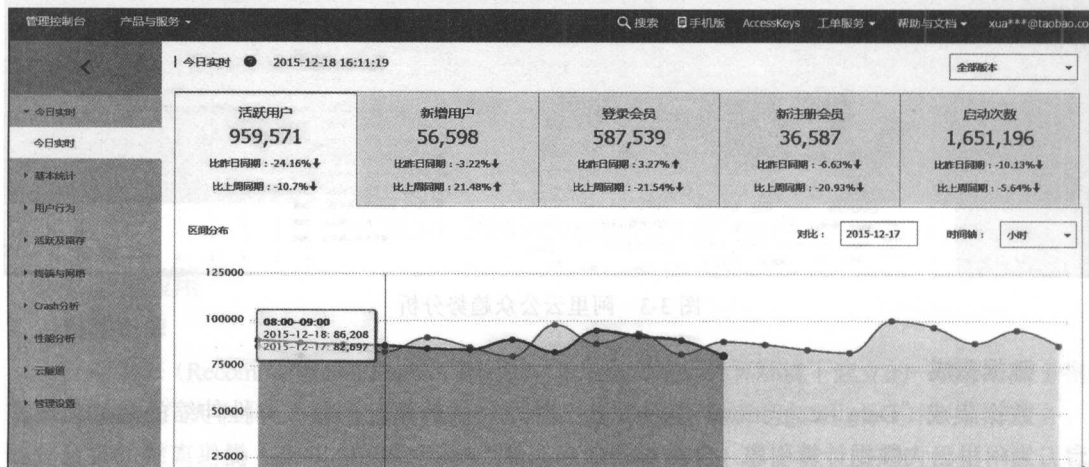


图 3-5 阿里云移动数据分析



提示 阿里巴巴还有另外一套移动数据分析产品——友盟，目前友盟已经跟纬元信网络数据、CNZZ 三家公司合并并统一更名【友盟+】，提供包括针对网站和 App 的统计分析服务，针对游戏、广告和线下分析的行业数据解决方案、自助智能分析，针对微社区、分享和推送的运营工具全域运营指数和运营报告。

数据市场相关 API 及应用

阿里云允许自身以及第三方企业通过 API 的形式提供大数据应用服务，它的定位是软件交易及交付平台，作为 2016 年阿里云的战略发展点，承接着中国云生态各个链条产品的落地。目前，入驻云市场的优秀 ISV 遍布国内外，提供围绕云计算产品的软件应用及服务，包括但不限于基础软件、服务市场、行业软件、企业应用、建站市场等。如图 3-6 所示为阿里云数据市场相关 API 及应用界面。

2. 数据分析展现

DataV 数据可视化

DataV 是一个可视化产品组件。相比于传统图表与数据仪表盘，其可视化致力于用更生

动、友好的形式，即时呈现隐藏在瞬息万变且庞杂数据背后的业务洞察。DataV 提供指挥中心、地理分析、实时监控、汇报展示等多种场景模板来帮助客户解决设计难题，并通过多种图表、数据源接入、图形化操作方式满足开发和设计需要，最终在终端适配多分辨率与发布方式，满足不同场合下的使用。如图 3-7 所示为阿里云 DataV 数据可视化界面。

首页	全部商品	
<p>完成质量 4.3分</p> <p>工作速度 4.1分</p> <p>服务态度 4.3分</p> <p>在线客服: 数据服务</p> <p>在线时间: 9:00-18:00</p> <p>电话: 1585811551</p> <p>邮箱: changji.sdm@taobao.com</p> <p>《服务保障承诺》</p>	<p>默认排序 上架时间 价格 评分</p> <p>印刷文字识别-身份证识别 (公测中)</p> <p>身份证识别服务可以自动地从图片中定位身份证图片区域, 识别出其中包含的身份信息。</p> <p>使用人数: 2416</p> <p>评分: 3.6分</p> <p>¥ 0.01</p> <p>人脸识别技术-人脸关键点提取</p> <p>定位人脸位置同时定位人脸上的一些关键点坐标。</p> <p>使用人数: 776</p> <p>评分: 暂无</p> <p>¥ 0.01</p> <p>印刷文字识别-行驶证识别 (公测中)</p> <p>机动车行驶证是准予机动车在我国境内道路上行驶的法定证件。-行驶证识别可以通过图片识别技术返回行驶证中的信息。</p> <p>使用人数: 651</p> <p>评分: 暂无</p> <p>¥ 0.01</p>	

图 3-6 阿里云数据市场相关 API 及应用

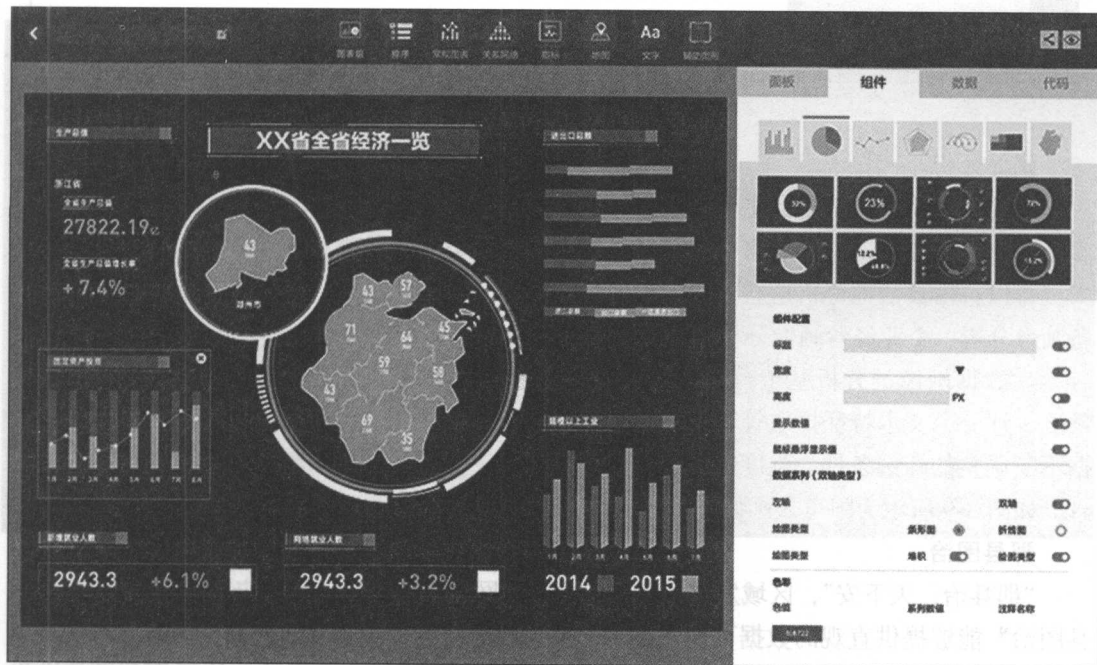


图 3-7 阿里云 DataV 数据可视化

QuickBI

QuickBI 是一个大数据商业智能套件，提供海量数据实时在线分析、拖拽式操作、丰富

的可视化效果，帮助客户更快地完成数据分析、业务数据探查。该产品更多地侧重于通过快速的数据整合、分析和可视化的方式提供简易可操作的数据分析服务。

QuickBI 内置柱状图、线图、饼图、雷达图、散点图等 20 多种可视化图表，可通过类似于 Excel 的操作方式进行多维数据分析；整个分析过程都是实时的，支持 RDS、MaxCompute (原 ODPS)、AnalyticDB 等多种云数据源；通过智能加速引擎针对海量数据提供秒级响应。如图 3-8 所示为阿里云 QuickBI 开始界面。

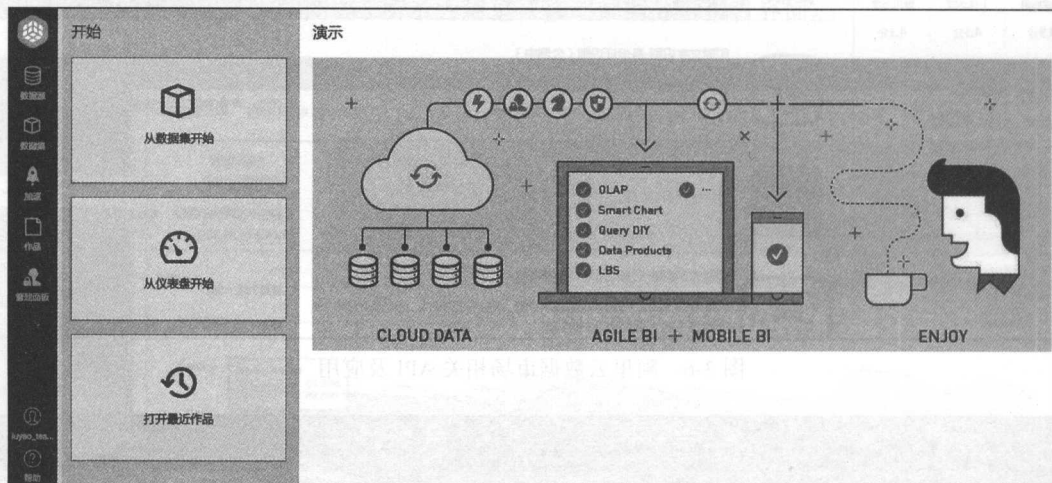


图 3-8 阿里云 QuickBI 界面

画像分析

画像分析所适用的场景主要是结合阿里云分析型数据库 (Analytics DataBase)，将分布在多个存储资源的数据整合起来，在标签模型上构建大数据画像类的交互式分析应用，让业务人员可以自由灵活地分析这些对象各种属性与行为之间的关联性。它可以广泛应用于工业设备画像分析、企业经营画像分析、用户行为画像分析等多个场景。

大数据画像类分析基于行为等明细数据产生，通过从半结构化数据中抽取特征并结合预测、评分、文本特征提取等算法技术来进一步挖掘有效用户特征。在交互式分析过程中根据不断调整的筛选条件、维度组合、下钻、上卷能够快速返回结果，直到获取到足够多的信息。如图 3-9 所示为阿里云画像分析。

郡县图治

“郡县治，天下安”，区域发展亟需响应“互联网”行动计划，敏捷应对经济新常态。“郡县图治”能够提供直观的数据可视化技术，整合政府统计数据和互联网数据源，动态反映当前区域经济的发展态势，集中呈现当地基础产业、特色产业、内需消费特征等各类关键指标，民生经济一览无余，为宏观决策提供分析依据和辅助支撑。“郡县图治”基于云计算环境部署，具备多种可配置参数，由阿里云实时推送互联网数据分析的结果，并提供全链路维护和自动化升级服务。如图 3-10 所示为阿里“云郡县图治”报告。

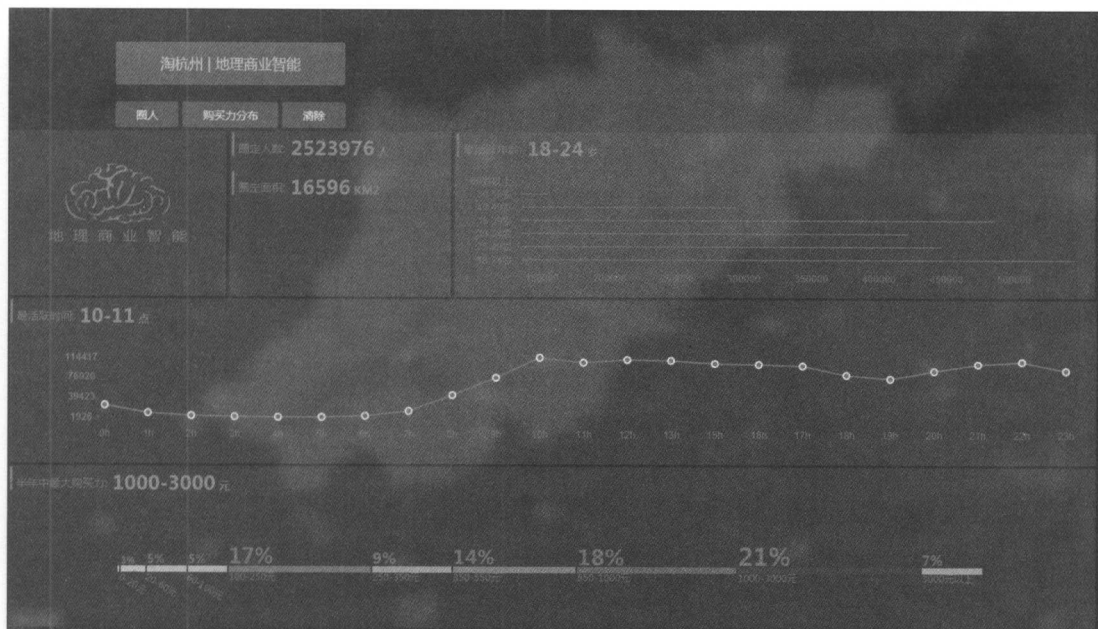


图 3-9 阿里云画像分析



图 3-10 阿里云“郡县图治”

3. 人工智能

机器学习

阿里云机器学习平台是构建在阿里云 MaxCompute 计算平台之上，集数据处理、建模、离线预测、在线预测为一体的机器学习平台。该平台为算法开发者提供了丰富的 MPI、PS、BSP 等编程框架和数据存取接口，同时为算法使用者提供了基于 Web 的 IDE 可视化实验搭建控制台。

它是一站式的算法与智能应用的开发、发布与分享的平台，所有工作都在一个平台上完成，减少了多平台转换、迁移、集成等繁琐问题；支持处理亿万级大规模数据，适用于绝大多数企业数据规模；基于工作流的思路，通过简单的拖拽即可完成数据挖掘、数据分析等功能。如图 3-11 所示为阿里云机器学习平台 workflow 操作界面。

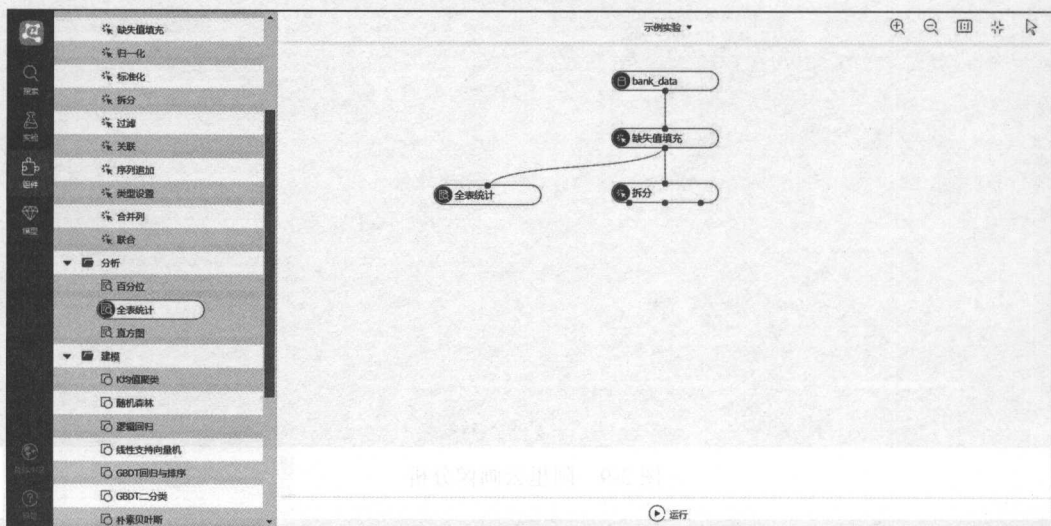


图 3-11 阿里云机器学习平台

智能语音交互

智能语音交互（Intelligent Speech Interaction），是基于语音和自然语言技术构建的在线服务，通过提供语音识别（ASR）、语音合成（TTS）、自然语言理解（NLU）为智能手机、智能电视以及物联网等产品提供“能听、会说、懂你”式的智能人机交互体验。

智能语音交互提供的三类服务：

- 语音识别（ASR）：它可以将语音转换成文字，支持多轨 WAV 格式的长语音文件识别、8k A-Law WAV、16k A-Law WAV、8k 16bit PCM、16k 16bit PCM 的格式，但目前语音只能识别普通话。
- 语音合成（TTS）：它提供的是将文字转换为声音的能力。
- 自然语言理解（NLU）：集语义解析、智能问答、意图识别等功能于一体，让应用具备理解能力。

印刷文字识别

印刷文字识别是 OCR（Optical Character Recognition，光学字符识别）的具体应用，它提供了包含多种场景下的文字识别，其中包括身份证证件识别、驾驶证识别、行驶证识别、营业执照识别、门店招牌识别、英文识别。

人脸识别

人脸服务是一款用于提供图像和视频帧中人脸分析的在线服务，通过提供人脸检测、人

脸特征提取、人脸年龄估计和性别识别、人脸关键点定位等，可应用于人脸美化、人脸识别和认证、大规模人脸检索、照片管理等各种场景。

通用图像识别

通用图像分析服务是一款用于提供图像内容分析和理解的在线服务产品。“通用”一词是指，在该服务中提供的算法 API 模块可以应用于各种图像领域，没有具体业务场景、垂直领域等场景限制。该服务旨在提供一些通用的图像分析和理解算法 API 模块，开发者和企业可以通过这些模块组合，结合自身领域特点，独立开发图像分离和理解系统，满足自身特定需求。

电商图像分析

电商图像分析服务是一款用于提供电商平台环境下的图像分析的在线服务产品。该产品提供若干图像分析和理解技术的在线 API 服务给开发者和企业使用，其中包括牛皮癣图像识别、图像背景分析、炒信图像识别等独立服务模块。这些独立技术模块可应用于电商平台下的商品主图、副图等质量判断、选品投放过滤、搜索和推荐等业务场景。

机器翻译

机器翻译 (Machine Translation) 通过阿里巴巴的海量电商数据，结合机器学习、自然语言处理技术，实现多语言语种识别与自动翻译功能，为跨境电商信息本地化与跨语言沟通提供精准、快捷、可靠的在线翻译服务。

4. 大数据基础服务

大数据开发套件

大数据开发套件 (Data IDE) 是阿里巴巴集团推出的大数据领域平台级产品，它提供了一站式大数据开发、管理、分析、挖掘、共享、交换等端到端的解决方案，其利用 MaxCompute (原名 ODPS) 在几分钟内可将原始数据转变为业务洞察的海量数据处理能力，整个过程都是通过对可视化组件的拖拽来实现。如图 3-12 所示为阿里云大数据开发套件配置界面。

数 管理控制台

工单服务 | 帮助 | youn****@hotmail.com

创建数据开发项目

提示：数据开发工具需要关联ODPS项目，如果您有阿里云ODPS资源，可以直接关联，如果您未购买ODPS资源，请先购买。点击[这里](#)获取帮助信息。

*项目管理员: youn****@hotmail.com

*项目名称:

如果您已经有了ODPS项目，请填写ODPS名称，系统将为您创建同名数据开发项目。
如果您暂时还没有ODPS项目，系统将为您创建数据开发项目和同名ODPS项目。

项目描述:

请填写您的AccessKey信息 (我不知道AccessKey信息，请前往阿里云控制台查看AccessKey信息)

*Owner AccessKey ID:

*Owner AccessKey Secret:

图 3-12 阿里云大数据开发套件

大数据计算服务

大数据计算服务（MaxCompute，原名 ODPS）是一种快速、完全托管的 TB/PB 级数据仓库解决方案。MaxCompute 主要服务于批量结构化数据的存储和计算，可以提供海量数据仓库的解决方案以及针对大数据的分析建模服务。MaxCompute 已经在阿里巴巴集团内部得到大规模应用，例如：大型互联网企业的数据仓库和 BI 分析、网站的日志分析、电子商务网站的交易分析、用户特征和兴趣挖掘等。如图 3-13 所示为大数据计算服务界面。

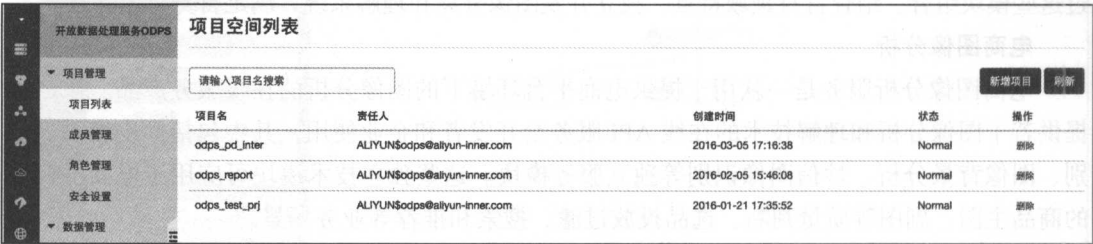


图 3-13 阿里云大数据计算服务

分析型数据库

阿里云分析型数据库（原名：分析数据库服务 ADS），则是一套 RT-OLAP(Realtime OLAP，实时 OLAP) 系统。在数据存储模型上，采用自由灵活的关系模型存储，可以使用 SQL 进行自由灵活的计算分析，无需预先建模，而利用云计算技术，分析型数据库可以在处理百亿条甚至更多量级的数据上达到甚至超越 MOLAP 类系统的处理性能，实现百亿数据毫秒级计算。如图 3-14 所示为阿里云分析型数据库使用界面。



图 3-14 阿里云分析型数据库

批量计算

批量计算（BatchCompute）是一种适用于大规模并行批处理作业的分布式云服务。Batch

综上，联合开发方式更适合那些想要对数据、技术完全可控，并且将大数据战略作为重要发展战略和核心竞争力的企业，这些企业通常内部已经具有一定的技术实力、较多的行业专家、相对明确的数据规划和预期。

3.2 如何选择解决方案

企业选择大数据解决方案时，需要综合企业外部环境、企业内部环境、需求规划、解决方案特性和解决方案费用评估分析五个方面。

3.2.1 外部环境分析

1. 行业情况

企业对于行业情况的分析，侧重于全面掌握未来几年或十几年的发展预期，大数据解决方案对企业决策会产生重要的影响，其对行业情况的分析主要包括：

- ❑ 业务增长规模。不同的业务规模对大数据解决方案的要求也有差异。比如，每日 200 万销售额跟每日 2 亿销售额完全是两个级别，后者则对大数据解决方案的整体架构、实时计算、批量查询、在线联机分析等能力要求较高。
- ❑ 预期业务布局。业务布局导致的行业特性变化会影响大数据方案的组建，尤其对应到顶端应用层会增加对特定垂直行业的经验要求。对大数据解决方案提供商而言，专注于特定行业意味着具有聚焦的技术和业务能力，也更利于为特定业务做出辅助决策或形成驱动效应。
- ❑ 市场角色地位。企业自身的市场地位不同，对应到大数据层面的市场战略分析、竞品分析的需求和关注点也不一样，这就对大数据解决方案提出了更高的战略性支持需求。
- ❑ 客户分析。企业的目标客户由自身的经营方向决定，目标客户可能包括 B 端（B2B，企业对企业）、C 端（B2C，企业对个人消费者）、G 端（B2G，企业对政府）及其他模式组合。不同的客户群体有各自特殊的分析方法和建模应用方案，这对大数据挖掘、计算和应用层面提出了更有侧重性的要求。
- ❑ 业务模式分析。不同的业务模式会产生不同的数据，并对数据的生产、加工、计算和分析产生不同的影响。比如，同样是 To C 的业务模式，淘宝、百度、腾讯完全是三种不同类型的业务，淘宝侧重于围绕交易形成个人生态圈，百度侧重于建立个人信息重组，而腾讯则连接了个人的社交以及围绕社交的生活。这些模式要求解决方案具备特定的采集、分析、挖掘、计算和应用支持。

2. 竞争对手

在企业选择大数据解决方案时，往往存在一种选择“惯性”——看看竞争对手在用什么，尤其是对行业领头羊以及跟自身相近的竞争对手这两类对象的选择更具参考价值。为什么企

业会非常重视竞争对手的解决方案选型？有以下四个方面的原因：

- ❑ 规避风险的需要。很多大企业在做方案选型时的一个重要出发点是最大限度地降低项目失败的风险，而竞争对手的选择会说明其他企业也有类似的案例，尤其是案例企业具有代表性，那么则意味着大家都比较认可该解决方案。
- ❑ 经验不足的参照。虽然在大数据实践方面有各自的差异点，但同一行业中的不同公司在业务上是相似的。企业对于自身大数据的差异点与大数据解决方案的匹配可能无法做出正确判断，但基于业务的相似性，企业可以借助于竞品选择来帮助自身做出快速且相对正确的判断。
- ❑ 增加方案的说服力。很多第三方服务商在做销售推介时，一定都会有一部分是关于销售和应用案例的。这些信息在企业进行内部提案时，是一个非常有效的增加方案说服力并推进落地进程的要素。
- ❑ 可供学习的模式。在某些情况下，如果行业内很多企业都选择同一个服务商，那么该服务商的产品或服务模式会基于该行业形成垂直型解决方案，这些解决方案其实是对行业经验的总结和共享。这有利于企业迅速了解该行业有关数据和业务的通用经验，提高自身大数据项目的实践能力和价值产出。

3.2.2 内部环境分析

1. 业务现状

企业内部进行业务现状分析的主要目的是了解现有业务对数据工作的认知、保障和约束，然后作为数据选型的基本出发点。

数据工作文化

数据工作文化是企业文化的一种，不同工作文化下，企业员工对于数据的价值认知、分析水平、数据结果理解和数据应用会产生不同影响。数据工作文化良好的企业由于在大数据工作方面已然形成工作机制，并且具备较高的数据工作技能，因此能借助成熟的解决方案提高大数据价值产出，即使是面对复杂的大数据解决方案也能有效加以使用；反之，大数据价值很难通过企业的工作文化融合到业务运营中，数据价值很难显现。数据解决方案的应用流程、应用场景、规范性要求、多部门配合机制、界面友好性、功能使用习惯、工具和文档语言限制等需要与当前工作文化相匹配。

团队组织架构

我们在第2章中已经介绍了企业大数据工作的相关组织架构和职能体系。大数据工作的不同环节需要具备特定技能的人才来实现。如果企业缺少特定的角色和组织保障，那么在选型中就需要注意这种缺失与方案落地实施的要求是否冲突。

数据工作能力

大数据工作不仅仅是技术部门和数据部门的事情，而且需要企业所有部门共同参与。不同部门的数据工作能力（例如数据理解、数据提取、数据应用、数据分析等）会影响大数据

解决方案的选择。通常，功能强大的大数据产品会提供较多的预置功能，另外也会提供灵活的自定义配置和部署能力，这些对于没有太多数据工作能力的人员来讲是一个非常大的挑战。因此，工具能否被有效利用，需要综合考虑现阶段人员的基本数据技能，以及掌握预选方案需要技能的上手时间和难易程度，同时还要考虑服务提供商是否有完善的培训、指导、应用和售后体系。

2. 数据现状

数据现状是对企业环境内可接触到数据基本情况的调查分析，包括数据源环境、数据结构类型、数据量级、数据质量、数据成长。

数据源环境

对现有数据源环境的正确认知是大数据整合应用的基础。数据环境的分析包括数据源的业务基础分为几个模块、各个业务模块和数据模块间的相互关系、数据存放的位置以及数据库（或数据文件）基本约束、不同数据库表和数据流转的元数据规范以及数据字典等。其中对选型影响最大的是源数据系统、异构复杂程度、同步更新信息、数据关联项等，这会影响到方案对数据源抽取、集成及后面所有的应用流程。

数据结构类型

大多数企业的数据都是以结构化的形式进行存储，而在很多业务环境中也可能存在大量的非结构化和半结构化数据，例如机器日志、报批文件、办公文件等。而某些情况下，业务数据既可能存储在结构化数据库中，也可能以半结构化甚至结构化的文件存储。对于这些信息的了解利于在选型时有针对性地考察大数据解决方案在特定方面的处理和计算能力，例如如果企业有语音数据，那么后期可能需要大数据解决方案能够实现语音文件抽取、存储、解析，以及针对解析后内容的语义分析和挖掘等。

数据量级

不同的数量级下，对应到解决方案工具本身的抽取、存储、处理和计算的能力要求也是不同的。同时，由于数据实时性的要求，在海量数据（例如 PB、ZB、EB 等）下，对大数据平台实时计算的要求更高。因此，在选择解决方案时，需要重点考虑工具对于海量数据在单位时间内的抽取、计算、建模、输出的能力以及实时性，并且考察当数据规模上来之后对系统压力、冗余性、安全性、并发性、响应性等的影响。

在企业数据量级的评估过程中，要综合所有可用的数据源。企业内部数据包括三类：常见的数据源都是业务类数据，包含业务运营的各个方面，例如采购、生产、库存、调配、物流、分销、营销、促销、客服等；除了业务数据外，企业的职能数据是企业内部运转的记载，包括人事、行政、检查、管理、计划、评审等职能线的数据；最后一类是来自于 IT 本身的日志记录，这是对 IT 机房环境、宽带网络、设备软硬件等机器运行数据（例如性能、应用、事件、错误等）的综合记载。综合这三类数据才能构成企业内部的完整数据。但是，企业不只有内部数据，还会通过多种途径与上下游产业链、渠道商、合作商等进行数据交换、整合和交易等，这些也是需要进入到企业数据量级的考虑范畴内。

数据质量

数据质量本身影响的不仅是在选择数据解决方案后的工作，更在选择方案之前就会对方案提出一定要求。在不同的数据质量下，对数据的加工、转换和处理要求是不同的。比如，对数据关联项明确、完整性高、同步及时、准确率高等的数据库，由于数据质量高对大数据工具的数据质量校验和处理要求会降低；但如果数据质量较低，那么会针对数据质量的各个方面形成较高的要求，比如，对数据完整性、一致性、及时性、准确性等方面的数据校验规则、数据质量度检查、数据异常处理、数据血缘分析、数据异动影响、数据关联分析等具有较高要求。

数据成长

随着企业的发展壮大，数据也在不断成长。在评估解决方案时需要有针对性地评估数据伴随业务增长带来的预期因素，尤其是数据量、数据种类、增长速度、数据计算需求、数据整合等方面。这些对解决方案的要求主要体现在产品的性能和功能扩展时的软硬件低成本、简易部署和运维、功能可定制开发、系统和组件的解耦、数据迁移成本等。

3. 制度要求

制度要求指的是企业对于所有运营工作的统一制度和规范。很多大型企业、国有企业、上市公司、外资公司等在这方面的表现明显，甚至能成为大数据方案选型的决定性因素。制度要求在数据方面的约束包括数据安全、主导权问题、数据所有权问题等。

数据安全

数据安全是企业开展大数据工作的基本前提。企业对于数据安全的要求会在数据存储环境、整合方法、关键字段加密、数据流通、人员工作环境等环节形成一定约束。比如，对很多传统银行来讲，关键数据不能流出数据中心是制度性要求，这在客观上要求数据方案要支持本地化或混合云的部署。再如，很多企业对关键字段的保密性要求非常高，并且即使在企业内部也会根据数据的安全性进行分区存储、处理和接入，这就要求整体解决方案中必须具备定制加密、分区隔离、流转控制等方面的能力。

主导权问题

企业大数据项目的牵头部门不同，那么在协调全部资源进行需求调研、项目规划、开发实施到最终交付的整个过程中都会有所偏重。比如，如果是业务部门（或偏业务类的部门）主导和推进，那么会更重视上层建模、分析、应用和落地的场景、模型等应用输出价值点；而如果是IT部门（或偏技术类部门）负责主导和推进，则对架构完整性、扩展灵活性、运维低成本、技术先进性、组件解耦性、系统兼容性等方面更重视。两种不同类型的业务体系在推进过程中的重视会影响最终方案的选择倾向。因此，如果是企业级的大数据项目，在负责总体调控的核心小组中，应该通过对核心领导小组成员的组织结构、利益构成、知识组织、技能要求、经验模式、行业配比等有效控制来降低这种偏向性选择风险。

数据所有权

与数据所有权问题相关的解决方案模式是云服务方案模式。在第三方云服务模式下，数据采集、跟踪和存储都是在云端进行的。此时，云端的所有数据是否可以完全被企业所有，

并且保存到企业内部是很多重视数据所有权的企业关心的问题。这一问题不仅关系到数据安全，更关系到企业未来数据资产。因此，如果企业对数据所有权非常重视，那么必须要选择本地化或可以将云端数据完全同步到本地的解决方案。



提示

云端数据同步到本地，常用的方法包括 API、文件下载等，这些方式可能存在请求或下载次数的限制，且不太适用于海量数据尤其是大文件类传输，海量原始数据更多地会采用 FTP 的方式将数据文件传送到指定服务器上。但对于海量数据而言，这种方式受制于发送端服务器、接收端服务器、网络宽带等的稳定性、传输效率的影响，经常会由于传输中断、网络不稳定、服务器权限、发送服务问题等出现数据残缺、损坏、不完全等问题。除了这些问题，海量原始数据以及处理后数据的传输实时性也是企业的关注点，很多云服务提供商可以通过一定的方式对原始数据进行实时同步，但对于处理后的数据同步则会存在一定的延迟，这种延迟根据处理的复杂度和数据量级可能延迟到以“天”为单位的时间。

3.2.3 需求规划分析

评价一个整体解决方案是否合适，在成本规模的制约下往往更侧重于与需求的匹配度，而非功能的全面性。根据企业发展阶段以及数据工作文化的不同，企业的数据需求和规划会存在很大差异，但总体上包括企业转型需求、业务应用需求和技术工作需求三类。

1. 企业转型需求

在企业发展的不同阶段，尤其是增长面临困境时，企业可能面临着转型的需求，转型过程中可能涉及数字化运营、个性化服务、流程模式重构、组织结构重组等内容，此时需要大数据在各个方面发挥辅助决策甚至应用驱动作用，是否具备针对这些内容的聚焦点和解决方法，是考察大数据解决方案的侧重点。

2. 业务应用需求

在数据应用端，应用大数据的对象包括企业自身、企业的目标客户、企业的合作伙伴甚至整个行业。企业需要根据自身情况通过调研总结得出具体需求，并考察解决方案的满足或偏差程度。如下是一些常见的业务性应用需求：

- ❑ 方案支持多少标签以及打标签的方法；
- ❑ 如何将已有的其他工具的数据计算或挖掘直接或迁移应用到现有平台；
- ❑ 方案具有哪些客户生命周期模型并如何对客户流失进行分析；
- ❑ 如何通过社会化媒体提取客户声量、口碑和满意度；
- ❑ 如何通过灵活的自定义配置新增或减少特定数据的跟踪采集，而减少对技术的依赖；
- ❑ 方案提供多少种数据挖掘模型；
- ❑ 方案支持哪些数据挖掘或机器学习库，是否支持第三方开源工具如 R、Python 等的算法库；

- 是否允许对特定维度定义灵活的预警规则并监控触发，是否可以将该过程自动化；
- 是否支持非代码类的数据 workflow？例如拖拽式工作方法；
- 方案中有哪些可应用到营销领域的分析和挖掘模型，都能得到哪些结论；
- 如何通过方案和工具来规范数据工作流程，并逐步建立数据工作文化；
- 方案是否可以基于文本字符串进行查询检索，例如在搜索框中输入“昨日有哪些业务线销售额异常变化”能直接得到对应的业务线名称、销售额以及变化量等。

3. 技术工作要求

在技术端，企业关心的问题既包括整体方案和架构等宏观的部分，又包括具体技术和开发细节的微观部分。如下是一些常见的大数据解决方案的技术型需求：

- 如何对多个数据源进行统一标记和采集，形成具备可整合和分析价值的高质量数据；
- 如何实现多异构、复杂数据源的数据拉通和整合；
- 如何实现全景数据的共享及分发；
- 如何对多地、不同公司主体间的元数据进行统一管理；
- 如何基于现有系统进行改造和升级，尤其是低成本、低风险、快速、安全的改造策略和方法；
- 如何通过统一的平台针对不同业务部门提供个性化、可定制的数据分析、应用功能，并减少产品冗余和降低二次开发成本；
- 如何兼顾技术平台的效率、性能、安全、成本、易用性；
- 针对常见的大数据工作，例如实时处理、交互性分析、数据挖掘、机器学习、离线批处理、海量数据 SQL 查询、数据可视化、商业智能、推荐引擎等，方案中的数据分析需求通过什么技术来实现，各自的优化点和增强点有哪些；
- 如何通过云服务实现针对企业不同国家、地域、体系来提供多租户、高可用、虚拟化、模块化、通用流程的灵活服务；
- 通过何种服务可对外提供数据管道、海量数据集成服务和数据输出服务等。

3.2.4 解决方案特性分析

1. 产品特性

对于大数据解决方案中的产品特征，重点考察产品层面的能力和特性，包括弹性付费、弹性配置、方便扩展、方便管理、简单易用、灵活控制、功能丰富、海量数据支持、简易实施、数据安全、可迁移性、运维成本等。

- 弹性付费。弹性付费是针对具有弹性 IT 需求的一种灵活的付费方式，弹性付费不仅可以提高大数据投入成本的利用效率，更能减少对财务支出成本的压力。
- 弹性配置。弹性配置是与弹性付费相关的特性，更多的是云服务的配置方式，支持弹性配置的工具更能满足企业不断变更的需求。
- 方便扩展。产品扩展包括整体服务器和集群扩展、服务器的硬件配置扩展、软件环境

- 功能升级以及组件和服务、应用场景的扩展等。
- ❑ 方便管理。大数据平台需要能将数据系统、业务系统关联起来，形成对数据、功能、流程、应用的全面管理；同时通过监控报表对数据主体以及应用数据的对象的行为进行监控。
 - ❑ 简单易用。由于企业内会有具备不同技能层次的用户参与产品应用，产品如果具备较好的易用性特征，则能更容易被所有人使用，也更利于数据价值的产出。
 - ❑ 灵活控制。面对复杂的数据需求，产品需要能根据不同场景提供定制化应用能力，包括资源配置、数据管控、界面组织、功能配置、环境限制等。
 - ❑ 功能丰富。对于一款工具而言，其功能越丰富代表可通过工具获得的业务洞察越多。
 - ❑ 海量数据支持。大数据的特征之一就是数据量大，工具对海量数据（数据规模）的支持程度，尤其是处理效率、结果、性能等是重要关注点。
 - ❑ 简易实施。在所有技术相关的解决方案中，IT 部署实施是一项非常耗费人力的事情。如果解决方案中的技术产品能具备一键部署、管理、转移等功能，将大幅度降低实施成本。
 - ❑ 数据安全。本书已经多次强调了数据安全的重要性，工具对于数据安全的支持是企业考虑方案采购的重要维度。
 - ❑ 可迁移性。很多优秀的大数据解决方案在提供强大功能特性的同时，也使得企业一旦使用了这些功能后便会被其绑架而不得继续使用，后期在迁移时会导致数据无法导出、结构无法识别、格式不兼容等问题，因此可迁移性涉及后期系统升级换代和替换需求。
 - ❑ 运维成本。对于完全本地化的大数据工具，其本身的软硬件更新、扩展，服务授权，功能变更，危机故障处理等方面的成本也需要考虑。

2. 功能特性

功能特性是指大数据解决方案在技术方面的功能特征，包括基本部署、数据导入、数据存储、数据计算、机器学习、可视化、应用支撑、云服务、数据安全、运维管理等。

- ❑ 基本部署。支持基于 X86 的集群方式，支持通过私有云、混合云等方式提供大数据服务。
- ❑ 数据导入。支持 SQOOP、Goldengate、Canal、Java-API 等技术实现抽取过程，支持文件、结构化数据、JSON、流式数据等数据类型的抽取。
- ❑ 数据存储。支持结构化数据、非结构化数据的存储，支持 HBase、Hive、MongoDB、Redis、关系型数据库和图形数据库等，并可提供 PB 级以上应用服务的数据仓库。
- ❑ 数据计算。支持离线计算，例如 MapReduce、HiveSQL、ImpalaSQL、SparkSQL、RHadoop、RSpark、UDF（Hive UDF、Impala UDF）以及实时计算 Spark 或 Storm 等。
- ❑ 机器学习。支持监督式学习、非监督式学习、增强学习的各种算法，在实现组件或算法库上支持 Mahout、R、Python、MLlib 等开源机器学习工具及其中核心算法库的集成。
- ❑ 可视化。提供丰富的可视化图表，除了常规图表外还包括玫瑰图、桑基图、热力图、树图、网络图、平行坐标图等；支持对开源组件的集成，如 Echart、Hchart、D3 等；另外，可提供针对商用可视化工具例如 Tableau 的支持，同时可将报表嵌入到其他报表系统中。

- ❑ 应用支撑。支持通过 IDE、SDK、Web 等方式进行应用开发，支持无需编程的文件检索、数据查询、交互式分析、临时分析、拖拽式应用等；提供针对应用系统的接口或集成，例如个性化推荐、精准营销、智能客服、机器翻译等。
- ❑ 云服务。提供多租户的软硬件资源和数据隔离应用，提供计量计费功能，提供 JDBC、ODBC driver 等多种驱动，以 SQL 的方式访问大数据平台的数据。
- ❑ 数据安全。支持数据传输通道和数据加密等保密机制，企业级安全认证机制（例如 LDAP 等），以及 SSO 验证；支持数据表单元格级别细粒度分析验证；支持对关键数据透明加密，无需修改上层应用，同时加解密过程不会对性能造成影响；支持集中的密钥管理功能。
- ❑ 运维管理。提供基于策略的数据备份和恢复功能；提供图形化、免维护的安装工具及配置和部署工具；提供统一的集群监控分析功能，支持基于事务和事件的报警等运维管理工作；提供集群配置参数的多版本管理能力，查看具体的修改内容，并支持版本回退；提供 REST 编程接口，能够通过调用编程接口实现集群部署、角色分配、服务启动和停止等功能；能够实现业务在无中断的情况下进行软件版本的升级及打补丁。

3. 性能特性

针对不同的技术组件会有不同的评估指标，例如硬件类、存储类、计算类、Web 事务类、网络类、查询类等，评估指标主要集中在伸缩性、容错性、单位时间处理能力、响应时间、吞吐量、并发性、稳定性、资源占用率等方面。

- ❑ 伸缩性：伸缩性是一种对系统平台弹性计算处理能力的设计指标，它是考察平台对硬件的增减或不同规模下处理数据的自适应能力的重要指标。
- ❑ 容错性：容错性是指在故障存在的情况下计算机系统不失效，仍然能够正常工作的特性。它是系统在异常情况下能良好运行的重要保障。
- ❑ 单位时间处理能力：处理能力几乎是所有组件都需要考察的指标，针对不同的组件其处理能力需要综合平台的配置情况，处理任务包括读、写、扫描、排序、连接、聚合、复杂计算等。
- ❑ 响应时间：响应时间是从发出请求到得到响应的的时间。响应时间越短，对终端计算、应用的实时性和体验越好。
- ❑ 吞吐量：吞吐量指在一次性能测试过程中网络上传输的数据量的总和，它能说明系统级别的负载能力。
- ❑ 并发数：并发数指系统对同一事务同时处理的请求数。并发数越高说明系统对事务在同一时间下的并发支持度和宽容性越高。
- ❑ 稳定性：稳定性是系统在不同场景下运行的稳定效果，稳定性越好，其可适用的场景越广泛。
- ❑ 可靠性：可靠性是在一定时间内、一定条件下无故障地执行指定功能的能力或可能性，大多数平台都会保证至少 99.9% 的可靠性，或者每年少于几个小时的故障时间。

❑ 资源占用率：不同的服务都是基于底层软硬件资源的支持，在总体资源有限制的情况下，资源占用越少且又能保证平台的技术组件或服务越有优势。

如表 3-1 所示是某大数据产品中存储和计算部分的性能评估规格。

表 3-1 某大数据产品技术性能规格

类 别	指 标	规 格
并行计算引擎（MapReduce）	WordCount：平均每节点处理能力（GB/ 分钟）	6GB/ 分钟
	Terasort：平均每节点处理能力（GB/ 分钟）	5GB/ 分钟
并行计算引擎（Spark）	WordCount：平均每节点处理能力（GB/ 分钟）	25GB/ 分钟
	Terasort：平均每节点处理能力（GB/ 分钟）	4GB/ 分钟
Hive	处理能力 -HiveAggregation：平均每节点处理能力（GB/ 分钟）	6GB/ 分钟
	处理能力 -HiveJoin：平均每节点处理能力（GB/ 分钟）	2GB/ 分钟
HBase	100% 随机读：平均每节点读取记录条数（每条记录 1KB），响应时间小于 50MS	25 000 记录 / 秒
	100% 随机写：平均每节点写入记录条数（每条记录 1KB），响应时间小于 50MS	30 000 记录 / 秒
	顺序扫描：平均每节点 scan 记录条数（每条记录 1KB），响应时间小于 50MS	9 000 记录 / 秒

注：该测试结果基于如下配置：节点数：12，CPU：2×E5-2650，内存：128G，硬盘：SATA 盘。

4. 服务特性

大数据服务是当前大多数企业付费意愿较低的内容，原因是服务很难有明显且特别有价值感的落地交付物，它不像一个产品、一个报表那样可以直接以产品化的方式展示。但服务对于企业，尤其是刚进入大数据阶段的企业至关重要。大数据服务包括以下几方面：

- ❑ 实施部署。服务商通常需要完成大数据系统的搭建、调试、优化、测试，使之能支持企业客户基于大数据平台进行应用开发。
- ❑ 质保服务。在大数据工具交付之后，服务商需要提供一定时间（例如一年免费升级、三年免费故障解决等）的质保服务。
- ❑ 技术咨询。调试完成后，对安装、配置、调试的所有信息、验收文档、交付手册等向用户进行全面交接，并提供技术咨询。
- ❑ 驻场开发。在大数据工具实施和开发阶段，客户可能需要服务商驻场开发。
- ❑ 工具培训。服务商需要根据企业需求和大数据解决方案本身，就实施部署、后期运维、工具开发、产品应用等方面提供培训教材并作系统性的推广培训工作。
- ❑ 日常沟通。日常沟通的问题会涉及大数据解决方案的各个方面，支持的方式需灵活且多样（邮件、电话、进驻企业），对于沟通的效率同时也应该有所要求（2 小时答复、7×24 小时服务）等。
- ❑ 应急故障。对于由于服务商提供的大数据工具本身的问题导致的故障问题，服务商也需要提供针对性的响应机制，包括解决时间、解决策略、实施步骤、质量验收等。

除了上述4个特征需要针对性的分析外,针对大数据整体解决方案的整体架构、技术细节、产品增强点、产品创新点等也是需要评估的关键内容。

3.2.5 解决方案费用评估

大数据解决方案的费用,主要指的是方案采购本身,而不包括外部其他机房、硬件、人员、设备等的投入。大数据解决方案分为云服务和本地化两种。

1. 云服务费用

目前大数据解决方案中出现了多种云端服务模式,例如IaaS、OaaS、PaaS、SaaS、DaaS等。不同的模式对应到大数据平台收费方式也有所差异,对云端“解决方案即服务”类的费用而言,主要费用集中在云端服务本身。不同的云服务内容对于收费内容的定义主要侧重于两方面:

(1) 按服务配置项目或需求收费

对于不同云服务的模式,根据用户选择的不同配置情况以及使用的服务进行收费,适用于弹性用量以及需求变更较大的场景。如图3-16所示为阿里云存储服务计费的方法。

存储计费

存储到MaxCompute的数据,包括表(Table)和资源(Resource)等,会按照其数据容量的大小进行阶梯计费,计费周期是:天。MaxCompute以小时级别采集用户每个项目空间下在当前的存储使用情况,并以项目空间为基本单位,计算用户当天的存储平均值再乘以单价:

基础价格	大于100GB部分	大于1TB部分	大于10TB部分	大于100TB部分	1PB以上部分
0.0192元/GB/天	0.0096元/GB/天	0.0084元/GB/天	0.0072元/GB/天	0.006元/GB/天	请通过工单联系我们

如果任意项目的实际存储量小于512MB时,MaxCompute将收取这个项目0.01元/天的费用。例如:如果用户在MaxCompute上某个项目的存储的数据为100MB,MaxCompute不会按照上述阶梯价格计费,而是直接收取用户0.01元/天的费用;如果用户含有多个项目,MaxCompute会为每个项目收取0.01元。如果用户某个项目的存储为50TB,则每天收取的费用为:

1. $100\text{GB} \times 0.0192\text{元/GB/天}$
2. $+900\text{GB} \times 0.0096\text{元/GB/天}$
3. $+9216\text{GB} \times 0.0084\text{元/GB/天}$
4. $+40960\text{GB} \times 0.0072\text{元/GB/天}$
5. $=382.8\text{元/天}$

图3-16 阿里云某云存储服务计费

(2) 固定/包断收费

这是一种相对固定的收费方式,根据用户选择的套餐或服务按照一定周期固定计费。在该方式下服务的内容是有一定限量或限额的,适用于需求和发展规划相对稳定且明确的场景。如图3-17所示为阿里云针对电商的云端整体解决方案收费方法。

2. 本地化费用

本地化大数据解决方案的费用通常由多种内容组成,主要包括三大类:硬件费用、产品费用和服务费用。

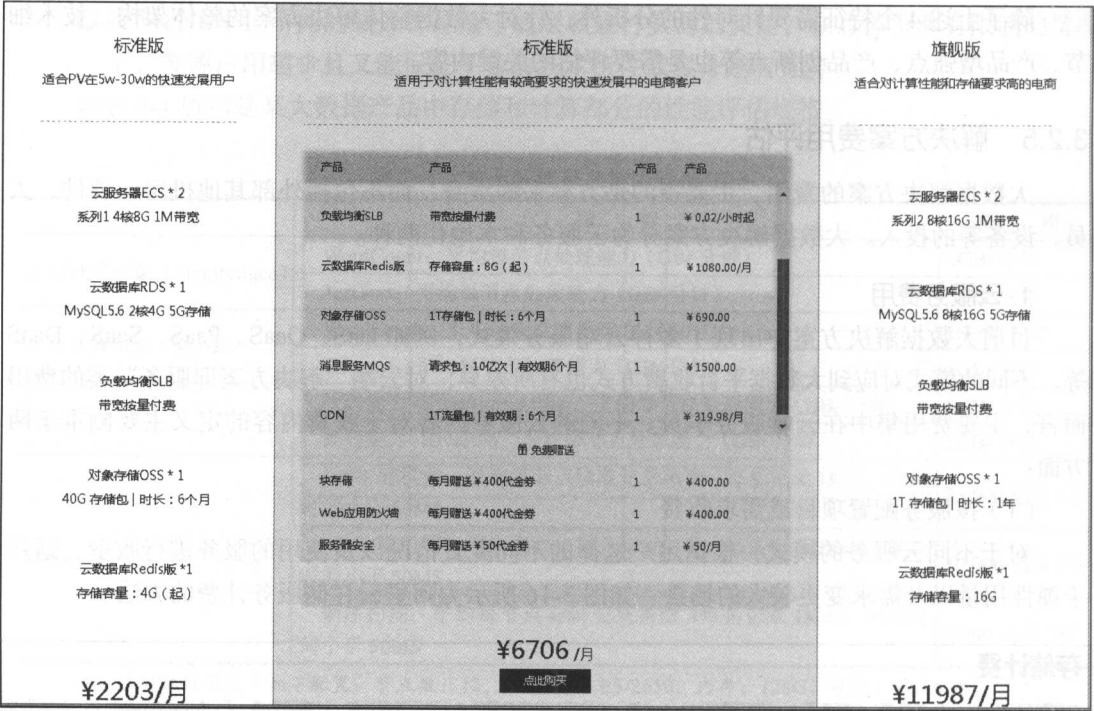


图 3-17 阿里云针对电商的云端整体解决方案收费

- ❑ 硬件费用：部分大数据厂商会将其解决方案与特定硬件做绑定销售，客观上这会增加企业前期购买和后期运维的额外成本；同时，这种“依赖式”的绑定也可能导致软硬件的一体化封装，对于后期的功能扩展、性能提升、安全防护等方面造成严重阻碍。
- ❑ 产品费用：不同的大数据解决方案中对于产品费用的定义方式是不同的，例如按 license 收费、按数据量和计算量收费、按节点数收费、按功能收费、按功能组件收费等，不同的收费模式对应的费用结构也不同；同时，对于按照功能类的收费意味着后期在需要应用某些功能模块时可能面临需要支付额外费用的问题。
- ❑ 服务费用：服务费用主要是人力资源类的费用，可能包括技术开发人力外包、特定人员驻场、后期使用培训、关键技术故障解决、应用场景和模型等方案，这些相对“不标准化”的费用组成与企业需求、实施难易程度、自身技术实力、后期运维实际、发展规划等有关。

大数据解决方案的选择，一定要结合企业现有状态、需求规划（包括短期、中期和长期）、预算、项目目标等，并综合考虑服务商的客观环境、产品、服务、预期产出价值等因素进行综合评估。强大的工具不一定适合所有企业，而且同一个工具也不一定适合于同一个企业的不同发展阶段。

综合上述所有内容，选择解决方案时各个因素重要性汇总如表 3-2 所示。

表 3-2 解决方案选择要素重要性汇总

类 别	子类别	细分内容	重要性评级
外部环境分析	行业情况		★★★☆☆
	竞争对手		★★★★★
内部环境分析	业务现状	数据工作文化	★★★☆☆
		团队组织架构	★★★☆☆
		数据工作能力	★★★★☆
	数据现状	数据源环境	★★★★☆
		数据结构类型	★★★★★
		数据量级	★★★★★
		数据质量	★★★★☆
		数据成长	★★★★☆
		数据安全	★★★★★
		主导权问题	★★★☆☆
	制度要求	数据所有权	★★★★★
需求规划分析	企业转型需求		★★★☆☆
	业务应用需求		★★★★★
	技术工作需求		★★★★★
解决方案特性分析	产品特性	弹性付费	★★★☆☆
		弹性配置	★★★☆☆
		方便扩展	★★★★☆
		方便管理	★★★★☆
		简单易用	★★★☆☆
		灵活控制	★★★★★
		功能丰富	★★★★☆
		海量数据支持	★★★★★
		简易实施	★★★★☆
		数据安全	★★★★★
		可迁移性	★★★★★
		运维成本	★★★★☆
	功能特性	基本部署	★★★★☆
		数据导入	★★★★★
		数据存储	★★★★★
		数据计算	★★★★★
		机器学习	★★★★★
		可视化	★★★★☆
		应用支撑	★★★★★
		云服务	★★★★☆
		数据安全	★★★★★
		运维管理	★★★★☆
	性能特性	伸缩性	★★★★☆

(续)

类 别	子类别	细分内容	重要性评级
解决方案特性分析	性能特性	容错性	★★★★★
		单位时间处理能力	★★★★★
		响应时间	★★★★★
		吞吐量	★★★★★
		并发数	★★★★★
		稳定性	★★★★★
		可靠性	★★★★★
		资源占用率	★★★★☆
	服务特性	实施部署	★★★★☆
		质保服务	★★★★★
		技术咨询	★★★★☆
		驻场开发	★★★★★
		工具培训	★★★★☆
		日常沟通	★★★★★
		应急故障	★★★★★
			★★★★☆
解决方案费用评估	云服务费用		★★★★☆
	本地化费用		★★★★☆

3.3 本章小结

本章介绍了企业实践大数据方案的三种途径：独立研发、直接购买第三方解决方案和联合开发，并详细阐述选择不同解决方案时应该考虑的因素，并对各个因素的重要性进行评价。本章的所有知识点都非常重要，尤其是在选择解决方案时应该考虑的 5 个维度以及各自关注点，其中重点内容如下：

- 重视企业外部环境中的竞争对手分析以及内部环境中的制度要求；
- 掌握业务应用需求以及技术工作需求，并能根据产品、功能、性能和服务特征加以匹配。

企业大数据自主实施思路

当前,新一代信息技术与传统企业的深度融合(互联网+),正在引发影响深远的产业变革,形成新的生产方式、产业形态、商业模式和经济增长点。各行各业都在加大科技创新力度,基于云计算、大数据、人工智能、物联网、智能设备的互联网技术正在引领制造方式变革;个性化定制、供应链优化、财务预测、营销管控、库存优化、用户全生命周期管理、智能客服、电子商务等正在重塑产业价值链体系。

当企业决定实施大数据战略时,应该如何开始?本章将围绕企业大数据在自主实施之前如何规划进行讲解,包括规划原则、目标蓝图、建设目标、组织规划、技术方案、人才规划、投入产出评估、数据风险管理几个方面。

4.1 制定规划原则

在企业级大数据应用系统设计规划过程时,为确保系统的建设成功与可持续发展,在系统的建设与技术方案设计时我们遵循如下的原则:

4.1.1 价值性

公司在做大数据规划之前需要弄清楚一些问题:到底有多少数据?数据都是什么类型?数据分布在哪些环境?数据质量到底如何?这些数据都能干什么?数据价值如何提取?很多企业中的业务部门想针对一些数据进行分析时,却不知道能拿到什么数据,这直接导致了无法准确描述数据价值预期的问题。所以企业开始设计大数据系统实施方案之前,应首先由数据部门牵头,对企业现有业务系统的数据情况进行全方位的梳理,了解各系统的运行情况和

各系统之间的关系，完善各个系统的数据字典，并结合数据字典对原有系统中的数据质量进行评估，形成系统数据质量提升方案。

对数据价值性的评估是建立大数据平台的前提和原始动力，如果一家企业自身业务系统数据不够完善且数据质量低，说明数据价值度也低，首先要做的工作是完善业务系统数据规范，提升数据质量，这样才能在大数据系统实施后真正做到数据价值最大化。

4.1.2 实时性

在传统的数据挖掘统计中，不管是数据标签还是数据模型，一般都是通过数据库或传统建模工具定时执行生成的，这种处理方式有几个弊端：

- ❑ 只有当事件发生一段时间之后，通过数据报表才能看到，数据延迟性较大；
- ❑ 发现问题时再去补救已经为时已晚，需要花费大量的时间和资源去做数据和业务的回滚；
- ❑ 在数据仓库创建的过程中，它必然要根据业务系统数据的更新而进行迭代，实现数据完整无误的增量更新，是传统数据仓库建立时最大的技术难点；
- ❑ 数据仓库在与业务系统对接后，更需要有效地保障业务系统实时读取和操作相关数据的能力。

所以，为了实现数据实时录入、海量数据实时计算、生成动态实时标签、数据实时提取投入应用等关键节点，需要在大数据平台设计时系统地解决这些问题。如果在大数据平台建设之后仍无法解决，那么大数据平台就沦为一个升级版的“传统数据仓库”，系统价值就会大打折扣。

4.1.3 高效性

当企业中的业务数据系统使用了一段时间后，通常会由于跨多平台和异构数据环境、海量数据的复杂计算、延伸业务模型的优化修正以及重复计算任务的冗余而导致大数据平台效率的低下。

另外，数据平台上线后，随着业务量的增加，原有的计算资源将面临严峻考验。如何根据计算任务的重要级别进行资源分配，使重要任务优先运行；如何解决传统 ETL 和数据挖掘模型少则几个小时，多则几天的运行效率；如何协调开发和调试阶段资源的分配都是需要解决的问题。

以上问题都是对大数据平台高效性的考验，如何最大程度保障平台执行效率、数据高效的整合能力、数据模型的计算能力、资源分配能力等，都是在平台设计和实施时必须要考虑到的。

4.1.4 安全性

大数据平台安全是由系统类、功能类、数据类、资源类四个层面组成的。一般情况下，

系统类、功能类、数据类安全是业务相关的，需要具体问题具体处理。而资源类相对来说比较独立，在服务端体现为 ETL、算法及服务器的运行权限，在客户端则体现为数据模型的使用权限。如何将权限分配给用户，不同的大数据集群拥有不同的授权模型，授权模型和组织机构模型有很大的关联性。考虑到企业大数据的共有特性，在整个规划中，我们需要从下面四个层次来了解大数据平台的系统安全：

（1）系统类

在客户端，系统类安全涉及访问 IP 段的限制、登录时间段的限制、连接数的限制、特定时间段内登录次数的限制等，为用户提供和其权限相关的用户界面，仅出现和其权限相符的菜单、操作按钮；在服务端，则对 URL 程序资源和业务服务类方法的调用进行访问控制，是大数据平台的第一道防护大门。

（2）功能类

功能类安全会对程序流程产生影响，例如用户在操作业务记录时，是否需要审核，上传数据文件不能超过指定大小，操作按钮可控制的功能范围等。这些安全限制已经不是对入口的限制，而是对大数据平台操作流程的限制，这在一定程度上会影响平台的运行。

（3）数据类

数据安全包括两个层次，其一是字段级数据安全，即用户可以访问大数据平台的哪些库、表、字段；其二是行级数据安全，即用户可以访问字段下的哪条数据。一般以用户所在角色或组为条件进行权限分配。

（4）资源类

从硬件和软件上对大数据平台的执行任务进行控制，用户通过客户端提交数据执行任务时，大数据平台根据用户的级别、任务的重要程度，自动为任务排序并分配 CPU、内存等计算资源，以便更好地利用有限的平台计算资源发挥更大的作用，集群的容量大小直接影响到任务运行的效率。

以上四个层次的安全，按粒度从粗到细的排序是：系统类、功能类、数据类和资源类安全。

4.1.5 延展性

在大数据平台设计的过程中，为了最大限度地增强平台的价值，最大限度地吻合各业务部门的需求，充分考虑平台今后的硬件扩展、功能扩展、应用扩展、集成扩展等多层面的延伸，整个实施过程也应该始终贯彻面向数据价值，围绕平台应用，依靠业务部门，注重实效的方针。保证平台的延展性可以提高稳定性且可靠度高，满足用户需求不断发展的要求，便于应用程序的升级及扩展，减少应用系统再开发（二次开发、定制）的工作量从而降低成本。一般地，我们可以从以下几个方面考虑：

（1）组件化结构

采用全组件化结构设计，每个组件都被独立地实现，并通过标准接口联系在一起。每个

功能组件在功能上独立，同时可根据用户需求灵活配置、组合，实现平滑升级扩容。功能实体可使业务和开发人员根据具体使用要求增加或减少系统应用模块。

（2）标准化接口

采用标准统一的接口设计，所有功能实体间的数据交换以及对其他模块的数据引用都通过标准接口完成，使多个组件对接时在开放性、稳定性、扩展性与集成性上有着很好的适配空间。

（3）开放的功能包

平台除了组件化结构设计与标准化接口设计以支撑开放体系结构外，为了方便用户个性应用的开发，还应该考虑封装平台及其组件所需的二次开发应用工具包，使其他技术团队对平台进行二次开发时能够更好地复用。

4.1.6 全局性

大型企业尤其是集团性企业通常具有非常多的业务群，要建立一套既能满足整体需求，又能适应各个子体的大数据系统，需要企业做好顶层设计。

顶层设计涉及大数据项目的各个方面，具体如下所示：

- ❑ 平台整体技术架构。整体设计大数据平台从底层到应用层的技术架构，包括数据源与数据接入、数据清理与提升、数据存储与检索、数据学习与挖掘、应用模型封装、服务层搭建等。
- ❑ 物理和虚拟部署架构。大型企业内部的数据环境往往涵盖物理设备与虚拟化设备，针对性的大数据部署架构也可能产生基于不同环境的对接。
- ❑ 软硬件资源评估。对大数据平台搭建涉及的需求以及开发所需要的各种资源需要整体规划，避免资源冗余和浪费。
- ❑ 整体组件和功能组成。对于大数据系统内部不同功能之间存在的技术、组件高效率复用，尽量实现功能和组件间的松耦合关系。对于外部其他系统之间的兼容性也需要纳入大数据系统设计之中，外部系统可能包括数据系统（例如报表展示系统、数据采集系统、虚拟化产品等），也可能包括业务应用系统（例如推荐系统、调度系统、库存管理系统等）。
- ❑ 平台公有云、私有云和混合云设计。对于大数据平台的实现可能包括公有云、私有云以及混合云三种场景，不同的实现场景都应该有相应的解决方案。最终的终端应用场景会在企业内部或外部，以产品化界面或功能服务或 API 等形式展现，因此这也意味着在设计之初需要考虑多种服务场景支持的可能性。
- ❑ 数据综合治理方案。从整体层面对数据进行全生命周期管理，包括数据标准化、元数据管理、数据安全防护、数据隐私与脱敏、数据质量评估与提升等。
- ❑ 应用整合与细分应用场景。所有的上层应用都应该在规划阶段做好，与之对应的底层或中间层的功能实现才能针对性的开发，进而可以避免需求改变导致之前的系统被整

体推倒或重新设计开发架构的风险。

- 平台可维护性与升级策略。对于大数据平台建设完成之后的可维护性包括硬件可方便扩展、软件可自动化部署、不间断的升级及补丁修复、集群整体监控与界面化管理、服务的高持续性和可用性、平台高执行效率、低成本维护和升级方案等。
- 项目实施前后的培训和内部推广。对于大数据系统的实现需要企业内外部各个部门和公司的支持，因此对于内部的大数据价值、应用等方案的引导和推广非常重要。尤其是当大数据系统完成并交付之后，直接落地应用的是一线的各个部门，因此针对各个部门的整体培训、指导甚至制度约束等工作必不可少。

4.2 制定目标蓝图

不同企业对于大数据平台的预期有所差异，但在制定目标蓝图时的原则和方向基本是一致的，下面以国内某大型烟草公司的目标蓝图为例进行说明。

国内某大型烟草公司大数据平台软件采购与实施项目的建设目标是基于平台即服务（PaaS）的理念，通过采购成熟的大数据处理平台，并结合 Hadoop 生态相关的技术，通过客户化设计，建设具有国内某大型烟草公司业务特色的企业级大数据服务平台。

该平台需具备以下几大能力：

- 大数据量、细粒度的数据处理能力——能支持 PB 级以上的数据交换、存储、运算和应用；
- 交换和处理多种类型数据的能力——具备处理结构化、非结构化数据的能力；
- 支持多租户服务的能力——用户通过平台能自助按需使用有权限的或脱敏的数据以及平台所提供的应用工具、算法和服务，同时支持生产运营和万众创新测试大数据应用；
- 支持软硬件的灵活挂载和管理的能力——平台本身具有较好的弹性，具备较强的容错能力和较高的系统运行稳定性和安全性；平台具备资源管理、监控等功能。

该平台需要具备以下特性：

- 分布式运行环境，可根据业务需要支持灵活的节点挂载，满足不同阶段平台运行的需要。
- 全方位的数据处理和存储能力，包括结构化数据、非结构化数据的处理、存储和应用的能力。
- 超强的大数据量、大数据容量的运算处理能力，满足大规模、超大规模数据量（PB）的处理和服务要求。
- 可根据国内某大型烟草公司的实际需求进行客户化定制，充分考虑企业内现有各种不同技术的可兼容、可覆盖，满足各业务源系统的数据、应用集成和分发需求。
- 全面的资源监控、管理、预警能力，满足对硬件资源、应用系统的全面监控和管理的需要。实现对硬件实体和虚拟资源的动态管理，可根据应用模块的实际需要，动态分配和回收系统资源，应用开发人员可完全摆脱对软硬件资源的关注和依赖。

- ❑ 平台具备面向企业内外不同层级用户的数据使用与服务能力。包括数据分析结果的直观展示呈现、数据分析过程的可视化操作、敏感数据的加密与脱敏处理等，能满足各种不同用户的数据使用要求。
- ❑ 平台自身具备较高的系统运行稳定性、安全性和超强的容错能力，平台支持 7×24 小时不间断服务。
- ❑ 该平台在技术手段上具备较大的领先性，提供标准的服务接口，供第三方平台的快速接入，同时提供开发的 SDK、IDE 等，加快应用的开发、测试和部署，适应敏捷项目开发。

最后，通过该项目的建设实施，需要建立健全国内某大型烟草公司大数据平台大数据服务体系的相关建设标准，包括大数据交换体系标准、大数据服务体系标准、结构化/非结构化数据存储与处理标准、资源管理体系标准等，为国内某大型烟草公司大数据平台的持续建设与优化打下坚实的基础。

4.3 制定建设目标

与建设目标相比，目标蓝图更加强调整体、远期和宏观的方向和定位，而建设目标更注重局部、近期（或阶段性）、微观的具体需求点。

就建设目标而言（如图 4-1 所示），对于企业级大数据的目标定位至关重要，因为它决定了大数据最后的成果以及价值，定位明确可以使整体实施进度提速，同时最终成果能够快速而有效地帮助企业提升最终业绩或实现企业的终极目标。从企业角度看，大数据功能主要用于服务客户、企业自身、合作伙伴和行业产业链。

服务客户	服务自己	服务伙伴	服务行业
个人用户画像 个性商品推荐 个性服务推荐 智能导购服务 商品周期推荐	销售数据助手 运营业务助手 营销数据助手 客服数据助手 财务数据助手	商品策略分析 商品定位细分 伙伴数据平台 店铺运营助手 店铺营销助手	行业发展趋势 品牌综合分析 品类综合分析 用户属性偏好 市场环境分析
特点：个性 便捷	特点：快速 高效	特点：简易 完善	特点：趋势 归类

图 4-1 大数据建设目标

（1）服务企业客户

在现有市场经济条件下，产能过剩在部分行业中的趋势愈发明显，商品重合度甚至服务的可复制性越来越高，然而面对客户需求不断变化的情况，我们需要通过大数据来满足客户的需求。个性化这个词在营销领域很早就被提及，但是直到互联网时代才具备逐步实现的条件，并最终延伸出了落地方式——千人千面。个性化不再局限于商品或服务营销上，还延伸

到客服、导购、客户商品或服务的周期性购买、客户画像等领域，全方位地通过大数据帮助客户满足其需求。与此同时，在这个科技迅猛发展和信息大爆炸的时代，客户所能接收的内容越来越多，这就要求商品或服务的使用需要在尽量短的时间内让客户快速使用，因此大数据下的客户需求特点是个性和便捷。

(2) 服务企业自身

为了适应客户需求的快速变化，抓住稍纵即逝的销售机会，企业在大数据方面的应用也要渗透到各个不同的职能领域。

- ❑ 销售助手，帮助业务人员及时了解客户的需求偏好、库存情况、供应商情况以及市场整体环境的影响因素；
- ❑ 运营助手，告诉运营人员，客户从哪里来，又到哪里去，他们想要什么，哪些商品或服务搭配组合能促进销售，什么样的活动既能提高销售业绩又能确保公司利润；
- ❑ 营销助手，让营销人员知道，客户从哪些渠道来，哪些渠道推广成本更低，效果更佳，渠道可优化的空间有多少，客户更喜欢什么样的文案或展现方式；
- ❑ 客服助手，使客服人员知道客户买了什么，投诉了什么，最近有什么样的诉求和需求，这样可以更好地服务客户，提升客户满意度；
- ❑ 财务助手，帮助财务或风控人员了解企业所存在的风险漏洞，主要的盈利来源，成本中心都有哪些，以及哪些产能可以进一步提升优化。

因此，在市场快速变化的前提下，要求大数据能够在第一时间告知相关的数据使用者，我们能做什么，要做什么和应该做什么，建立快速和高效的响应机制，是大数据服务于企业自身用户的特点。



图 4-2 分析不同年龄段人群的需求特征

（3）服务合作伙伴和企业产业链

为了更好地提升企业销售业绩，单纯依靠企业自身的努力还是不够的，如果可以借助合作伙伴的力量，让伙伴与企业共同成长，这样的企业才是伟大而长远的。如何整合合作伙伴的数据，并将数据实现与伙伴之间的共享和开放，打通企业上下游，更好地把控上下游的风险和机遇，这点对企业及伙伴也是非常有意义的。

大数据如何帮助伙伴也有典型的应用，例如商品或服务策略分析，帮助伙伴确定商品或服务的定位，以及如何扩大周边商品或服务的关联性，更好地服务企业本身；商品或服务的定位细分，帮助合作伙伴做细分市场，将商品或服务内容做细做精；还有最典型的店铺运营助手和营销助手，这是阿里巴巴提供的伙伴服务产品，为店铺提供更好更完善的数据分析和营销工具，助力店铺获取更多更有效的流量，提升店铺销售。由于合作伙伴的水平不同，对企业的诉求也不同，因此企业大数据在为合作伙伴提供服务或应用时，要求产品在使用上尽量简单，易上手，同时功能尽可能完善。

收集了客户行为数据、企业自身的生产数据、合作伙伴的数据，再结合竞争对手公开的数据或者企业自身在行业中的地位，整合各方数据资源后，在稳定的政治、经济和社会环境下，企业通过大数据的技术手段，可以针对行业发展趋势进行分析和预测，如针对某些品牌或者品类的综合发展分析，用户属性偏好的变化以及市场环境如何变化和调整，以此从中找到规律、方法和应对策略，引导企业、合作伙伴甚至整个行业健康有序的发展。

从企业宏观角度考虑，大数据可以是一个成本中心，同时大数据又可以是一个盈利中心。因为大数据建设是需要有持续的人员和软硬件投入，而大数据的良好运作在帮助企业提升经营效率，节约成本，创造业务收入的同时，还可以帮助客户、合作伙伴甚至行业获取需要的信息、优化产业流程，甚至带来收入，企业在这个过程也可以从中获取其他边际效益收入。所有的一切取决于企业对大数据目标的定位和服务的对象。在企业大数据的目标定位确认后，我们需要思考其组织职能应该如何确定。

4.4 明确组织规划

企业大数据建立和项目实施时的组织结构设计，是通过对人力资源的整合和优化，确立企业大数据自主实施和后续运营阶段最合理的管控模式，实现组织资源价值最大化和组织绩效最大化。狭义地、通俗地说，就是在人员有限的情况下通过组织结构设计提高组织的执行力和战斗力。组织结构设计的主要工作为：在大数据实施团队中，对构成组织的各要素进行排列、组合，明确管理层次，分清各部门、各岗位之间的职责和相互协作关系，并使其在大数据的实施战略目标过程中，获得最佳的工作业绩。从最新的观念来看，组织结构设计实质上是一个组织变革的过程，它是把大数据项目实施的任务、流程、权力和责任重新进行有效组合和协调的一种活动。根据时代和市场的变化，基于大数据的组织结构设计或组织结构变革的结果会大幅度地提高企业的运行效率和经济效益。

4.4.1 组织结构设计的作用

建立良好的组织规划,能动态地反映内、外部环境的变化要求,并能明确并协调组织中部门与部门之间的关系,人员与任务间的关系,使员工明确自己在组织中应有的权力和承担的责任,有效地保证组织活动的开展。它具有4方面的意义。

(1) 促进内部优化

- ☐ 公司各部门数据标准优化。
- ☐ 数据 OLAP 应用优化。
- ☐ 数据挖掘应用流程优化。
- ☐ 数据产品优化。

(2) 加强产业链上下游整合

- ☐ 会员需求与包销定制深度整合。
- ☐ 供应链预测与预警机制整合。
- ☐ 宏观战略与机会捕捉机制整合。
- ☐ 基于企业内部的云平台服务模式,包括数据分析类系统和数据应用类系统。

(3) 满足市场化导向和客户需求

- ☐ 实时获取和应用来自于顾客的数据信息。
- ☐ 通过数据体系发展满足于顾客需要的战略。
- ☐ 实施大数据战略对顾客的需求做出更好的预测。
- ☐ 针对外部合作伙伴的数据服务。

(4) 奠定企业高效运营基础

- ☐ 以数据为驱动,运营调节具有较强的针对性和快速灵活性,可根据需要快速调整生产和人员安排、改变生产计划。
- ☐ 有利于内部形成小型的高质量的专业化的团体(组),以及最优化的团体组合。
- ☐ 交易过程具有高度的透明性和开放性。
- ☐ 需求具有单向性(不可选择性)和相互依赖性。
- ☐ 客户进入线下店面的应用,基于用户入店后可识别标识有针对性地推送活动、商品和券等。
- ☐ 客户进入线上平台的应用,基于用户上线 Web/Wap/App 后识别唯一标识进行信息推送。

4.4.2 组织结构设立的导向

要确定大数据的组织职能,首先需要了解大数据都要做什么,都会做什么,以此为基础进行相应职能的设置和安排。从大数据的实施和应用上来看,大数据的工作架构可以分为四个层次,即源数据存储、数据整合、数据清洗和数据使用(如图4-3所示)。

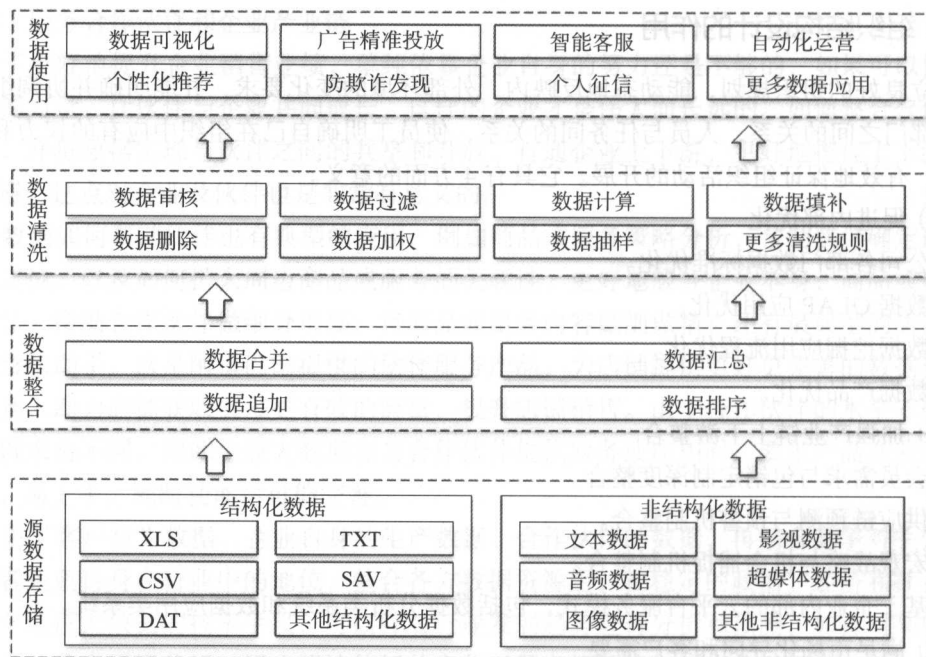


图 4-3 大数据运转流程

源数据指企业所拥有的数据，可以是企业内部数据和外部数据，其中内部数据包括本地数据（例如 FTP 文件、本地的 XLS、CSV、TXT、DAT、SAV、SAS、音频、图片和视频等格式数据）和各个应用系统数据（例如 ERP 系统、财务系统、物流系统、CRM 系统、网站页面和 App 等）。外部数据包含网络数据、合作伙伴数据和其他数据。巧妇难为无米之炊，企业拥有了数据源，才能正常开展大数据的实施和应用。

数据整合指将多数据源、多数据库、多数据表进行统一的规整，对应的数据源、库或表可以是结构化数据或非结构化数据，也可以称为关系型数据或非关系型数据。通过数据的整合，将原本分散在企业各个系统各个角落的数据，按照一定的格式或规则将其进行统一规范，并可随时进行查询、分析和调用。从整合的原理上看，整合包括数据合并、数据追加、数据汇总和数据排序等步骤。

数据清洗指按照一定的规则和使用系统的办法，将数据进行抽样、填补、删除、转换、筛选和计算的过程，令处理后的数据具备表达真实业务情况，并可以应用于日常分析、经营决策及战略方向的指导作用，同时形成既定标准的数据模型。清洗最重要的原则是清理规则和目标的制定，即在实际业务情景的指导下，为实现既定业务目的而开展的数据清理。因为只有在业务情景下的数据清理，最终才能与大数据的结果目标相吻合，才能使大数据被实际业务所应用，否则只是空中楼阁，中看不中用。

数据使用是大数据最顶层也是最终一层的目标，是将大数据的结果应用到企业生产或服务之中。从表现形式上看，包含数据的可视化和数据的非可视化；从数据的应用形式上看，

包含数据分析、数据模型、数据展示和数据智能；从数据的场景用途上看，包含智慧交通、智慧社区、智慧物流、智慧医疗、智慧企业、智慧银行、智慧政府、智慧家庭、智慧学校、智慧食品系统、智慧药品系统、智慧环保、智慧水资源管理、智慧气象、智慧企业、智慧建筑、智能楼宇、智慧农业等诸多方面；而若我们站在企业的角度考虑大数据的使用，则包含智能营销、智能财务、智能物流、智能客服、智能人力资源等方面。

4.4.3 组织结构的最终设立

鉴于大数据的工作架构，企业级大数据组织职能的理想结构如图 4-4 所示。

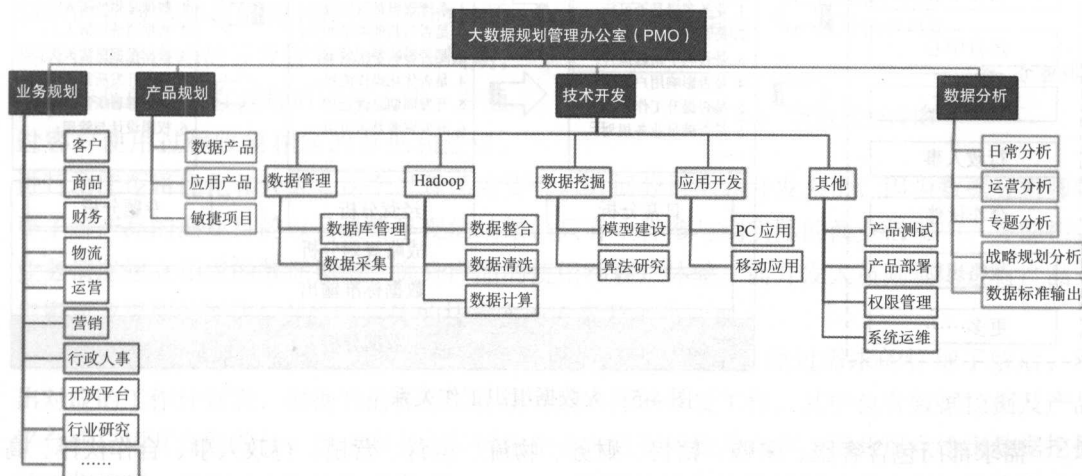


图 4-4 大数据组织架构

大数据规划管理办公室（PMO），负责协调所有与大数据项目相关的事务，包括大数据整体战略规划、产品的定位和输出、产品的应用价值评估及产品服务对象等内容。PMO 除负责整体的规划、监督和战略方向把控之外，还需要平衡资源分配以及对产品的最终价值负责，在企业里，所有不以商业目的为导向的行为都是没有价值和意义的，所有部门职能如此，PMO 也是如此。

另外，还要注意 PMO 不是业务线或 IT 线的一部分，而是凌驾于两者之上的机构。这种大数据组织必须由企业高层直接建立或采用自上而下的方式来部署。大部分国内企业在规划和实施项目时，都是从各个部门抽调人员，组成项目小组，而在项目结束时，各个部门的人员又回到各自岗位上，从而造成整个企业没有一个主动的项目推动点来推动围绕项目的闭环协作，而且没有持续的规划和执行。大数据的规划和实施同样也会遇到这样的问题，因此需要建立持续而稳定的大数据组织结构 PMO，才能督促其有效落地，同时针对企业的大部分情况，需要避免的是 PMO 只成为 IT 部门的事情。

除了 PMO 之外，还要建立以下工作团队：业务规划、产品规划、技术开发生态和数据分析，这四个团队是大数据的具体执行层面，对应大数据工作架构的四个层次，它们之间的工作关

系如图 4-5 所示。

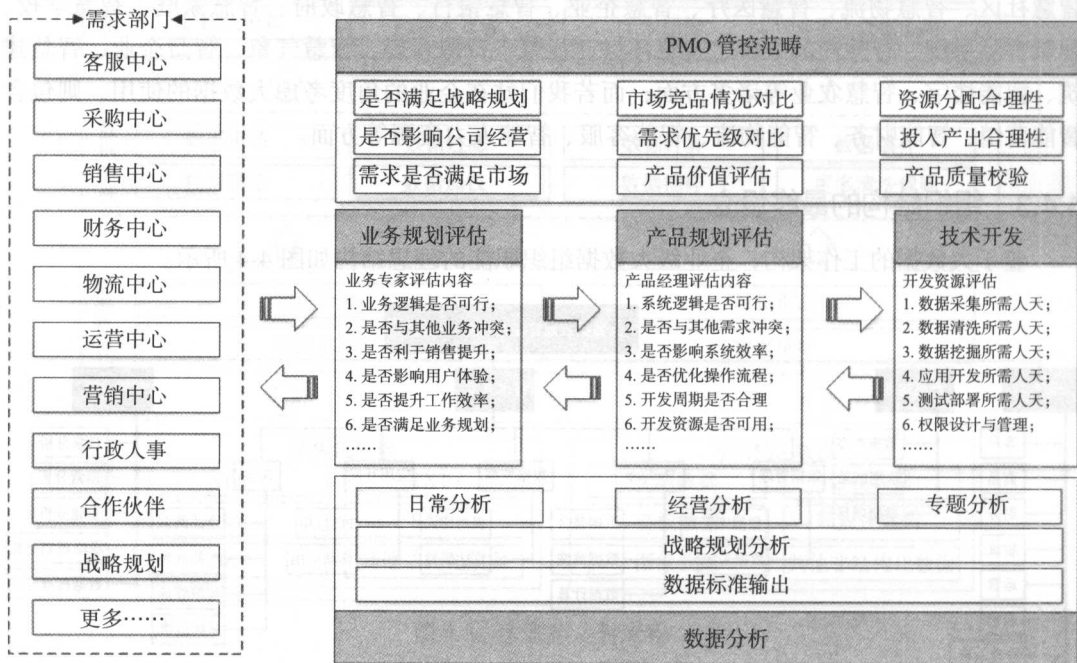


图 4-5 大数据组织工作关系

需求部门包含客服、采购、销售、财务、物流、运营、营销、行政人事、合作伙伴、高层管理甚至更多的其他职能，职能部门的需求只有熟悉职能业务的人才可以理解，因此需要将职能需求传递给业务规划人员进行评估。所谓的业务规划人员就是我们所说的业务专家，所谓的业务专家首选并不是从外部聘请或者从咨询公司受雇而来，业务专家都是从企业内部成长起来的，他们最了解行业，了解实际的业务内容、流程和规范，同时了解企业的过去、现状和未来，只有因地制宜的业务战略战术才最适合企业本身，才最有可能帮助企业快速发展和腾飞。如果企业规模较小或者没有培养自己的业务专家，此时可以聘请外部的行业专家结合公司内部业务领域的员工一同进行需求评估。

业务规划评估的角度主要是在体验、业务逻辑和销售帮助上。体验包括客户部分，同时也包含内部员工，也就是在进行需求评估时，需要考虑客户体验，以及内部员工及所对应的企业利益问题，只有这两方面都满足了才是最优的需求，但实际上不可能所有的需求都是最优的，因此我们在进行评估时一定要规避最差需求，尽量将次优和不提倡需求向最优需求进行优化和升级（如图 4-6 所示）。而业务逻辑则是保证是否能够顺利进行销售，降低错误率，提升销售效率，保证客户在购买商品或享受服务过程中能够更加舒适和自然，应该摒弃通过数据洞察发现客户的相关偏好和规律后，在提供销售或服务时使客户感觉被监视的错误应用。有了这些评估之后，还要考虑真正给销售带来的帮助到底有多大，如果逻辑过于复杂，造成体验降低，同时换取的销售额却很小，这样的需求可立即枪毙。

业务规划评估完成后，将需求信息向下传递给产品规划，产品规划则是从产品的系统性、逻辑交互、开发成本及周期的角度考虑。业务部门提交的需求与目前系统的兼容性如何；需要输入和输出哪些参数，参数是目前已经拥有，还是需要单独从哪些地方获取和计算；中间数据的计算和交互逻辑分别是什么，这个逻辑是否符合目前的数据标准规范，如果不符合规范，那么新的内容应该如何输出和使用；同时，还要评估需要占用的开发资源以及开发周期，督促产品的测试直到最后应用上线，并通过客户使用过程收集相应的意见和建议，进行功能逻辑优化和升级。这个过程还需要引入数据分析师和开发人员，因为数据分析师需要了解产品的商业目的以及最终呈现的结果，只有全程参与到实际的商业活动中，才能根据业务情况建立相应的数据模型，提升产品的使用和客户体验。而开发人员则辅助进行产品开发资源及周期的评估，并给出具体的技术实施方案及工作计划。

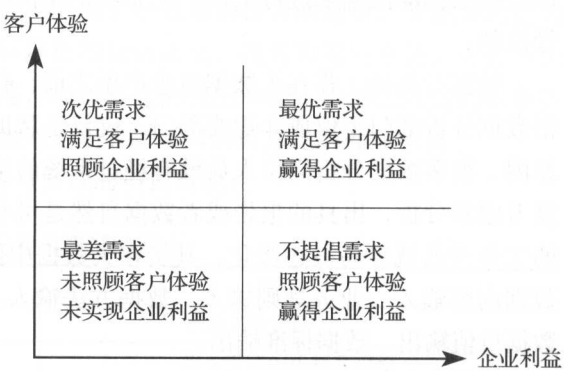


图 4-6 大数据矩阵图

在产品规划评估完成后，技术已经参与到了评估过程中，经过需求的详细了解后，给出对应的工作计划表，根据当前任务的排期开始具体的开发工作。其中包含数据挖掘及产品平台或应用开发的部分，鉴于本书主要讨论企业大数据的整体内容，因此重点给出数据挖掘的工作流程，如图 4-7 所示为根据 CRISP-DM（跨行业通用数据挖掘标准流程）的建模过程整理。

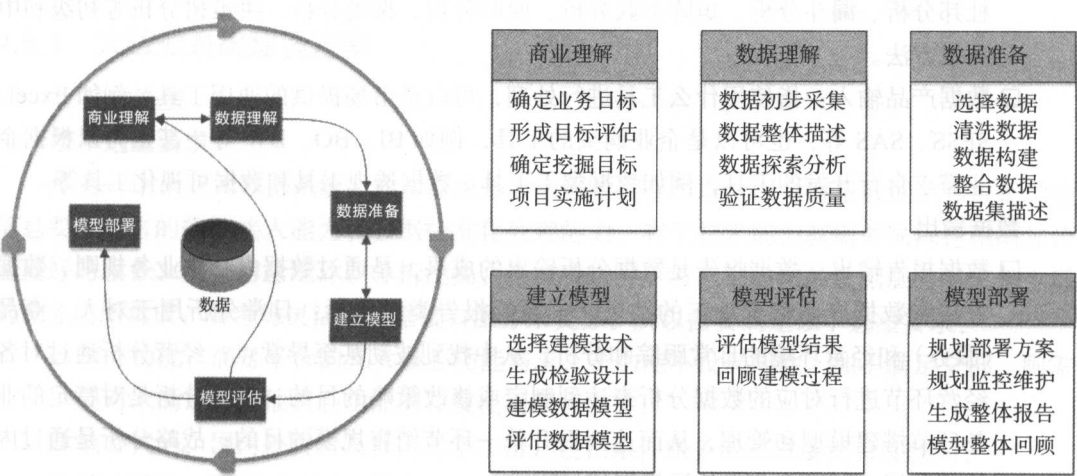


图 4-7 大数据挖掘流程图

其中，商业理解的部分就是上述业务规划评估和产品规划评估的过程，同时也是 PMO

对整体需求把控最核心和最重要的部分，因为只要把产品目标和价值确定好之后，后续基本都是根据标准流程来进行工作，无非就是在执行过程中，加入任务进度管理和结果质量的监管控制。

数据分析的工作在大数据概念产生之前，往往都是被忽视的部分，主要原因是大家容易把数据分析部门定位成非重要部门，仅仅是帮助做一些数据的统计工作而已，也是因为这个原因，很多数据分析岗位人员没有经过系统的培训，导致看待问题太片面，不能从全局的角度考虑和分析，出具的报告或者数据自然是可有可无或者起不到真正的商业价值，没有价值的工作当然就容易被边缘化。其实在大数据中数据分析的职能定位是输入输出，输入包括：数据内容输入、业务规则输入、数据方法输入、数据产品输入；输出包括：数据报告输出、数据价值输出、数据标准输出。

数据输入

- 数据内容输入：指数据分析师需要处理的对象，包括数值数据和文本数据，数据来源可以是内部系统导出、自己记录、部门间传递的数据等，还可以是外部第三方提供的数据、网络抓取的数据或其他公开的数据。
- 业务规则输入：单纯的数据处理不能得出任何有价值的报告或内容，需要有业务规则的数据，没有业务规则会导致统计指标的名称一致，但是结果却千差万别，因此需要确定清楚业务规则和目的才能确保指标的可用性。比如，用户的复购率这一指标，首先需要业务定义什么是复购，多长时间的购买算作复购，什么样的用户购买才计入复购，复购的是金额、件数还是订单数，复购对应的分母是什么。
- 数据方法输入：有了数据和规则，还需要选择正确的数据统计方法，否则也会导致出具错误或有偏差的结论，常用的数据统计方法包括：对比分析、分组分析、交叉分析、杜邦分析、漏斗分析、矩阵关联分析、回归分析、聚类分析、决策树分析等初级和中级方法。
- 数据产品输入：指使用什么工具进行处理，可以是市场提供的通用工具，例如 Excel、SPSS、SAS 等，也可以是企业购买的工具，例如 BI、BO、BW 等，甚至可以根据企业需求自行开发的工具，例如数据加工工具、数据透视工具和数据可视化工具等。

数据输出

- 数据报告输出：数据报告是数据分析输出的成果，是通过数据内容、业务规则、数据方法和数据产品整合加工的结果，主要的报告类型包括：日常分析用于对人、商品（服务）和经营环境的日常跟踪和分析，从中找到波动甚至异常点；经营分析通过对各经营环节进行对应的数据分析来达到制定或修改策略的目的；专题分析是对特定的业务环节搭建模型和管理，从而达到提升单一环节销售规模的目的；战略分析是通过内外部数据，制定企业中长远规划的过程。
- 数据价值输出：通过数据报告的输出，对具体的业务环节、业务链条、企业生态，乃至行业的发展制定明确的道路，并将报告结果应用至上述环节后产生了实际可量化的

经济效益。

□ 数据标准输出：数据分析部门不仅是数据的加工、处理和报告的输出，更重要的是为企业制定和留存具体的数据标准，包括数据指标的定义，报告数据的来源，处理方法和标准，数据输出规则和报告模板规范，通过多次的沉淀和修正后形成既定的标准方法，在提升数据分析效率的同时快速地为企带来可用和可量化的商业价值。

为方便读者理解，表 4-1 列出了数据分析与烹饪的项目对比。

表 4-1 数据分析与烹饪的对比

项目名称	数据分析	烹 饪
输入	数据内容	要加工的食材
	业务规则	添加需要的佐料及用量
	数据方法	料理的操作过程
	数据产品	加工用的锅碗瓢盆刀叉勺
输出	数据报告	成品的各种菜肴
	数据价值	菜肴或食谱换来了收入
	数据标准	烹饪的标准食谱

上述为企业大数据应该具备的基本职能和工作内容，由于各个企业的情况不同，需要按照企业的实际情况，在此基础上进行调整和细化，令每一个版块充分发挥其作用，实现统一目标下的企业大数据。

4.5 设计技术方案

4.5.1 大数据系统建设方案

1. 技术选型

技术选型要求

作为一个企业大数据系统，核心需求是大数据量的存与取，因为海量数据、多数据类型信息要有丰富的数据接入能力和数据标准化处理能力，有了技术能力就需要纵深挖掘附加价值更好的服务，例如信息统计、分析挖掘、全文检索等。考虑到面向的数据服务对象是上层的业务应用集成，要考虑灵活的数据接口服务来支撑，所以需要满足以下选型要求：

- 核心功能：满足平台的几大核心功能需求，子功能不设局限性。如不满足全部，则需要对未满足的其他核心功能的开发使用服务支持；
- 核心代码：国内外资料及社区尽量丰富，包括组件服务的成熟度流行度较高；
- 核心组件：需要对选型组件自身所包含的核心功能有较为深入的理解，易用其 API 或基于源码开发；
- 核心技术：商业服务性价比高，并有空间脱离第三方商业技术服务；

- ❑ 核心需求：一些非功能性需求的条件标准清晰，例如承载的集群节点、处理数据量及安全机制等。

技术选型原则

- ❑ 简单性：亲自试用大数据套件。这也就意味着：安装它，将它连接到你的 Hadoop 安装，集成你的不同接口（文件、数据库、B2B 等），并最终建模、部署、执行一些大数据作业。自己来了解使用大数据套件的容易程度——仅让供应商或外部顾问来为你展示它是如何工作是远远不够的，需要亲自做一个概念和应用验证。
- ❑ 广泛性：该大数据套件是否支持广泛使用的开源标准——不只是 Hadoop 和它的生态系统，还有通过 SOAP 和 REST API 服务的数据集成等。它是否开源，并能根据你的特定问题易于改变或扩展？是否存在一个含有文档、论坛、博客和交流会的大社区？
- ❑ 特性：是否支持所有需要的特性？Hadoop 的发行版本（如果你已经使用了某一个）？是否包含你想要使用的 Hadoop 生态系统的所有部分？是否包含你想要集成的所有接口、技术、产品？请注意过多的特性可能会大大增加复杂性和费用。所以请查证你是否真正需要一个非常重量级的解决方案。你是否真的需要它的所有特性？

技术选型陷阱

- ❑ 付费类：某些大数据平台采用数据驱动的付费方式，也就是说，你得为自己处理的每条数据付费。另外，还有以 License 收费的方式，例如定制服务需要额外收费。基于这些付费形式的产品或接近方案将会变得非常昂贵。
- ❑ 可控类：市场上所有大数据平台厂家都不会提供组件源码，实施方如果想进行二次开发只能通过厂家提供的底层接口来进行，要做平台升级只能通过厂家提供定制服务，这就意味着诸多底层权限、代码和方案的不可控。
- ❑ 兼容类：并不是所有的大数据平台都会基于 Apache Hadoop 接口规范进行封装，甚至有些厂家会定义特殊的规范，通常做平台切换和系统对接时，各系统上运行的程序代码互相兼容工作会变得异常繁杂，导致某些系统与其他系统无法实行对接，产生系统孤岛的情况。
- ❑ 应用类：还要考虑使用大数据平台的真正意图。某些解决方案仅支持将 Hadoop 用于 ETL 来填充数据至数据仓库，而其他一些解决方案只提供数据处理、转换或 Hadoop 集群上的大数据分析，当安装了冗余的大数据组件之后，平台的性能会受到很大的影响。所以真正需要多少 Hadoop 组件需要根据大数据的实际需求而确定。

2. 基本方案

步骤一：集群硬件优化

- ❑ 内存：结合用户的实际需求配置进行测试，达到最优内存配置。
- ❑ 硬盘 IO：结合用户的内存实际配置对运算框架进行调优，减少不必要的 IO 读写，提高 IO 读写效率。
- ❑ JVM 调优：选择合理的 JVM 内存配置。

通过既有的硬件环境，以及业务数据情况，对大数据平台的内存、IO、硬盘使用等各方面进行优化，使资源配置更为合理，资源利用更为高效。

步骤二：软件优化

在 MR 运算或 Hive 分析过程中，最典型的莫过于发生数据倾斜。如何在运算或分析之前预防数据倾斜，有效地提高运算或分析的效率，是实施团队在大数据平台实施过程中需着手分析并解决的问题。

基于目前业务应用使用情况的分析与统计，至少需要从以下几个方面进行软件客户化设计方面的优化：

□ 预防关联字段、聚合字段非空值。

□ 设置合理的 Task 数量。

□ 减少 Job 数量。

□ 提前合并小文件。

步骤三：后台系统数据迁移

在集群安装调试完之后，需要将原 Hadoop 集群的数据迁移到新的平台中，在迁移时需要考虑迁移的效率、数据的稳定传输，做到高效迁移，零数据的丢失。在 Hadoop 集群迁移过程中：

□ 应该剔除无用的数据，避免浪费磁盘空间，例如回收站的数据。

□ 如果使用分布式复制，因为是 Map 任务拷贝，因此要注意控制 Map 的数量。

□ 两个集群间的数据迁移要注意版本的兼容性，如果不兼容可以使用 WebHDFS 协议替代 HTTP 协议。

□ 为确保数据迁移安全，对迁移的脚本加以打印日志方式跟踪。

□ 为保障数据平滑一次性迁移成功，需预先对目标磁盘做空间与目录预估，避免因磁盘或目录相同导致一次迁移失败。

□ 要求保障投标方开发的程序平滑地迁移，极大减少资源废除，以及再开发的工作量。

步骤四：传统数据仓库的数据抽取导入

集群安装调试完之后，需要将原数据库中的 DW 数据整体迁移到新的平台中，由于涉及的数据量大、迁移的时效长，若想在迁移的同时又不妨碍平台正常运行，需要做合理的风控规避：

□ 采用 Sqoop 脚本后台运行方式抽取数据。

□ 抽取数据按照 DW 表的主维度抽取，不要一次性抽取全部数据，这样一旦出错中断，需要重头抽取，浪费时间，降低效率。

□ Hive 端建表时应统一采取建外部表，将数据直接抽取在外部表的指定目录下，然后在脚本抽取完之后，按照原 DW 表的主维度作为分区装载到 Hive。

□ 为确保 DW 数据迁移安全，对迁移的脚本加以打印日志方式跟踪，做到出了错误有迹可查。

□ 为保障数据平滑一次性迁移成功，需预先对目标磁盘做空间与目录预估，避免因磁盘

或目录相同导致一次迁移失败。

□ 要求保障源投标方开发的程序平滑迁移，极大减少废除，再开发工作量。

步骤五：业务系统数据的迁移

平台提供对企业所有业务系统的数据导入，包括可视化系统数据、营销系统数据等所有业务应用系统数据的导入，以及后续各业务系统的增量数据的导入。每个业务系统的导入应单独进行，互不影响，需要提供每个业务系统抽取导入的全量及增量模块：

□ 采用 Sqoop 脚本后台运行方式抽取数据。

□ 抽取数据按照日期时间维度抽取，不要一次性抽取全部数据，这样一旦出错中断，需要重头抽取，浪费时间，降低效率。

□ Hive 端建表时应统一采取建外部表，将数据直接抽取在外部表的指定目录下，然后在脚本抽取完之后，按照日期作为分区直接装载数据。

□ 增量数据的抽取以后台定时任务方式运行，按照日期作为分区导入 Hive。

步骤六：平台服务与二次开发

平台服务与二次开发通常都是基于 Web Service 进行的，这是一个平台独立的、低耦合的 Web 应用程序，可使用开放的 XML 标准来描述、发布、发现、协调和配置这些应用程序，用于开发分布式的互操作的应用程序。平台需要提供基于 SOAP 协议标准的 XML 文件数据格式的远程访问接口，用于外部环境调用用户平台的内部许可的数据，能使得外部环境不需要使用第三方软件或硬件，就可相互交换数据或集成。依据 Web Service 规范实施的应用之间，无论它们所使用的语言、平台或内部协议是什么，都可以相互交换数据，为企业甚至多个组织之间业务流程的集成提供了一个通用机制。

步骤七：多租户数据服务接口集成

多租户技术或称多重租赁技术，是一种软件架构技术，它是探讨与实现如何在多用户的环境下共用相同的系统或程序组件，并且仍可确保各用户间数据的隔离性。越来越多的企业都想将所有传入的数据保存在 Hadoop 中，因而它们的用户能够使用多种方式与这些数据进行交互：批处理、交互式、实时数据流分析等。而且更重要的是，它们要能同时执行这些交互，而不会出现在交互时单个应用或查询占用集群所有资源的情况。

需要支持以多种格式分析 HDFS 和 HBase 中数据的能力，并且不需要 ETL。用户可以使用多个框架来分析相同的数据。可以和 Map Reduce 一起运行在相同的物理机器上，支持企业的关键业务。支持对普通用户、管理员和 Web 提供了三种对外服务。

借助于 YARN 将 Hadoop 转化成一个多应用的数据系统，Hadoop 社区可以处理 Hadoop 所面临的新一代需求。YARN 在底层就满足了实际的需求。利用 Hadoop 的 YARN 组件，企业将能部署多租户的、服务于多个目标的 Hadoop 集群，这些集群可以满足不同组织和应用框架的各项 SLA 的要求。因此需要集成 YARN 组件。

多租户服务是企业大数据平台体系中一个面向各类不同数据和应用需求的数据服务平台，是大数据平台中支持正式运行、万众创新测试大数据应用的关键功能支撑模块。

该模块的重点是为大数据平台提供一个能快速接受用户原始数据，并能有效结合大数据平台本身已有的各种业务数据的功能，通过大数据平台提供的数据加工、处理、展示工具，根据用户的业务需要衍生计算出新的业务数据结果集，供用户对业务场景进行分析。

同时，在用户对数据结果集进行分析和验证后，用户可以通过大数据平台提供的管理工具，对该模块的数据结果、数据逻辑、数据调度关系提交注册。管理员在对该数据逻辑、注册内容审核后，进行数据和逻辑的发布。发布的内容包括数据结果集、调度逻辑和应用的权限。

内容发布完成后，拥有管理权限的用户，可以将该数据分析应用开放给企业内各类用户使用。通过一轮又一轮的用户自助性的数据加工、计算、分析、发布和分享，再分析，再应用和再分享，充分调动企业用户的积极性，充分挖掘企业大数据平台中数据的业务价值。

结合数据平台提供的资源管理和用户权限管理，进而实现平台的多租户能力，即在多用户的环境下共享相同的系统和程序组件，且仍然能确保各用户间的数据、配置及计算资源的隔离性；也能根据用户历史资源占用情况，适时调整资源配额。

步骤八：数据查询检索接口集成

数据检索主要针对在线交互式查询，平台应该提供平台内所有结构化数据的查询 SQL 以及 JDBC 接口，并且能够接近 ANSI 的 SQL 标准，使业务系统查询 SQL 能够有效对接，并且为了实现更加有效的管理，应提供下列服务：

- 查询角色要求，不同的用户查询检索数据，拥有不同的表查询权限。
- 作为在线交互式查询，响应时间不应超过 5 秒，当查询结果量大时，超过 5 秒，应该加分页处理，避免长时间无响应。
- 查询返回结果应该提供导出模块。

步骤九：数据共享订阅服务接口集成

数据订阅服务指外部环境系统或用户因业务需要调用平台的数据而进行的操作，分为长期订阅、临时订阅两种方式：

- 长期订阅平台在设置的定时任务时间内向用户推送订阅的数据服务。
- 临时订阅是指用户或外部环境临时性地调用用户平台数据。

步骤十：提供系统元数据接口集成

支持开发各类元数据管理分析模型：

- 血缘分析：提供对数据输入输出的回溯过程查询。
- 影响性分析：该功能支持当某些实体发生变化或者需要修改时，评估实体影响范围。
- 元数据检索：提供元数据的多条件查询检索功能。
- 元数据浏览：提供元数据的浏览、查看、下载等功能。
- 元数据维护：提供元数据的基本增、删、改、查功能。

3. 成果案例

下面是某大型国有企业客情评分画像（见图 4-8 至图 4-13）。

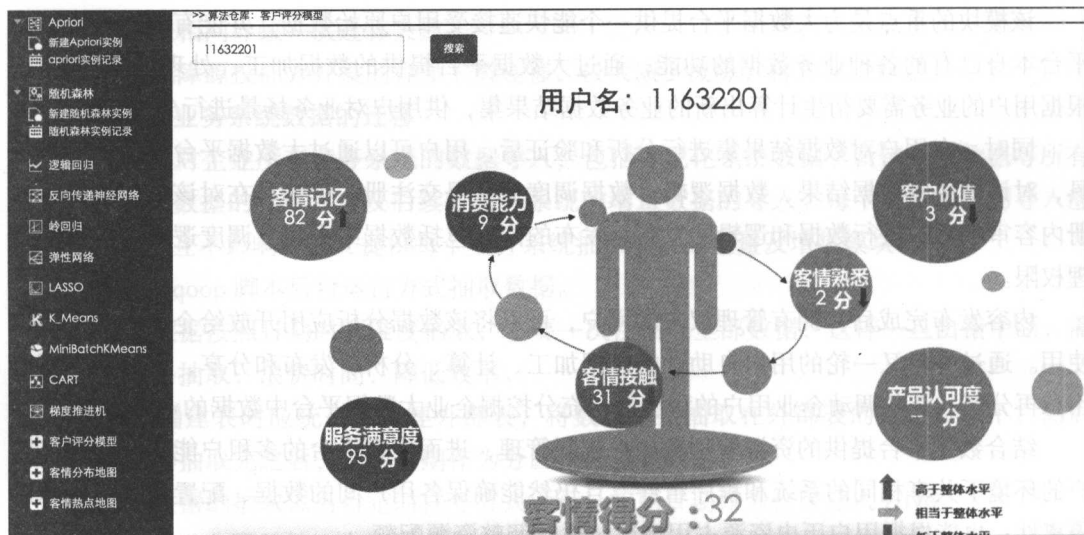


图 4-8 客情评分模型

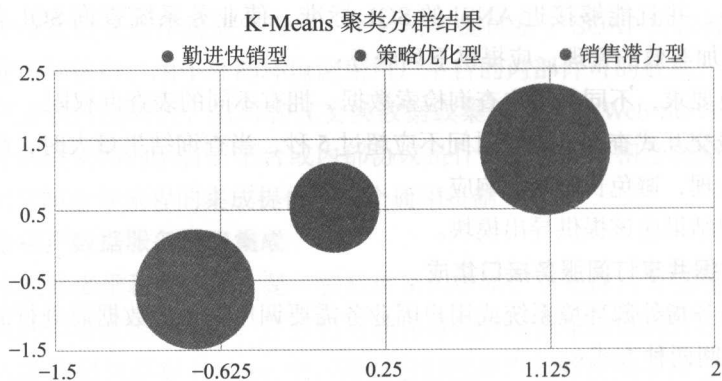


图 4-9 用户聚类模型 1

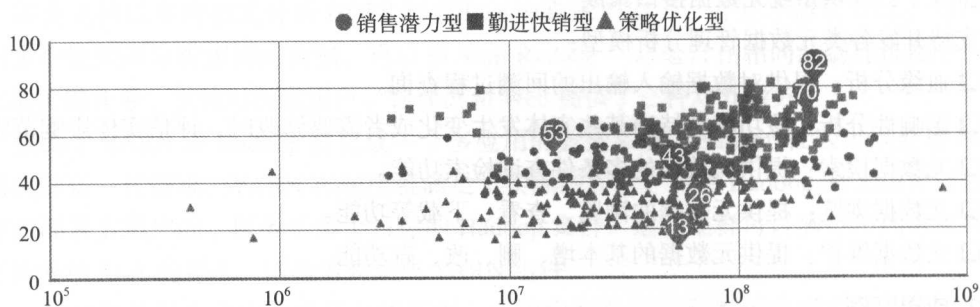


图 4-10 用户聚类模型 2

某大型零售企业大数据营销分析模型

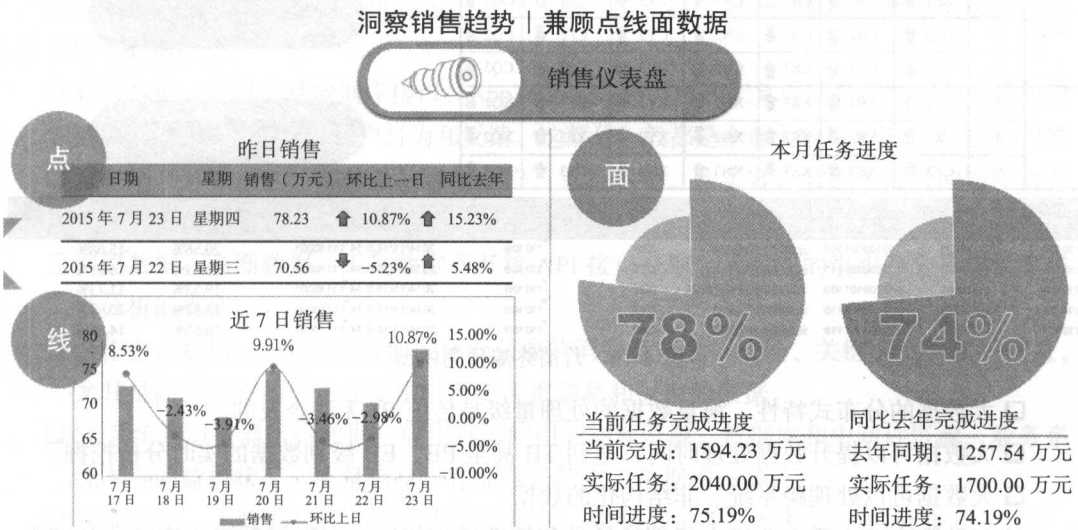


图 4-11 销售仪表盘



图 4-12 销售成因模型

4.5.2 大数据系统与传统 BI 的融合方案

在一般大型企业大数据系统前期建设评估的过程中，都会碰到一个无法避免的问题：新的大数据系统和原有 BI 系统的关系定位，是完全独立、资源互补还是全融合？行业内提到大数据时都会联想到 BI，对大数据关注度高的也基本都是 BI 的从业人员，然而大数据有很多区别于传统 BI 的典型特征：

市场类型	经营规模	食杂店 (Z)	便利店 (B)	超市 (S)	商店 (N)	烟酒商店 (Y)	娱乐服务类 (F)	其他类 (Q)
城镇	大	CZ1 ↑	CB1 ↑	CS1 ↑	CN1 ↑	CY1 ↓	CF1 ↑	CQ1 ↑
	中	CZ2 ↑	CB2 ↑	CS2 ↓	CN2 ↑	CY2 ↑	CF2 ↓	CQ2 ↑
	小	CZ3 ↓	CB3 ↑	CS3 ↑	CN3 ↓	CY3 ↑	CF3 ↑	CQ3 ↑
乡村	大	CZ1 ↑	XB1 ↑	XS1 ↑	XN1 ↑	XY1 ↑	XF1 ↑	XQ1 ↓
	中	CZ2 ↑	XB2 ↓	XS2 ↑	XN2 ↓	XY2 ↑	XF2 ↑	XQ2 ↓
	小	CZ3 ↑	XB3 ↑	XS3 ↓	XN3 ↑	XY3 ↑	XF3 ↑	XQ3 ↑

查看全部下降客户

查看全部上升客户

区县代码	上级机构代码	客户代码	客户名称	客户分类代码	营销区域代码	更新时间	省公司代码	环比上月	同比去年	更多字段
SALE_REG_CD	ORG_CODE	CUST_CODE	CUST_NAME	CUST_TYPE_CODE	SALE_REG_CD	UPDATE_TIME	V_ORG	RATE_LASTMONTH	RATE_LASTYEAR	NOTE
110109	11110001	110109101107	北京军庄腾源商店	XZ2	110109	2014/11/18 0:14	11110001	-20.56%	-15.68%
110109	11110001	110109101111	北京金特亿商贸中心	CY2	110109	2014/11/18 0:14	11110001	-20.13%	-16.57%
110109	11110001	110109101109	北京燕华恒通综合商店	CZ2	110109	2014/11/18 0:14	11110001	-19.53%	-11.23%
110109	11110001	110109101116	北京南成兴昌商店	CY2	110109	2014/11/18 0:14	11110001	-19.43%	-20.68%
110109	11110001	110109101118	北京惠泽吉利超市	CS2	110109	2014/11/18 0:14	11110001	-18.55%	-14.56%

图 4-13 营销终端预测模型

- ❑ 大数据的分布式特性，海量数据的处理量级是传统 BI 无法企及的。
- ❑ 大数据可以提升数据处理时间，达到 TB 甚至 PB、EB 级别数据的实时分析挖掘。
- ❑ 大数据可以处理跨系统、非结构化的数据。
- ❑ 和动辄百万、千万元以上级别的硬件和软件投入相比，大数据系统的建设成本也是一大优势。

和大数据相比，传统 BI 系统由于更早深入企业，结合业务形成了相对完善的数据报表体系，在现有业务的理解和呈现上比大数据系统更好一些，重要的数据指标也已经有所规范，所以对于大数据项目和 BI 的关系，建议短期以资源互补的方式运行起来，长期慢慢整合成一套完整的大数据业务和报表平台体系。

下面以案例的形式讲解某大型企业的大数据建设方案（如图 4-14 所示），其中包含与传统 BI 整合的部分。

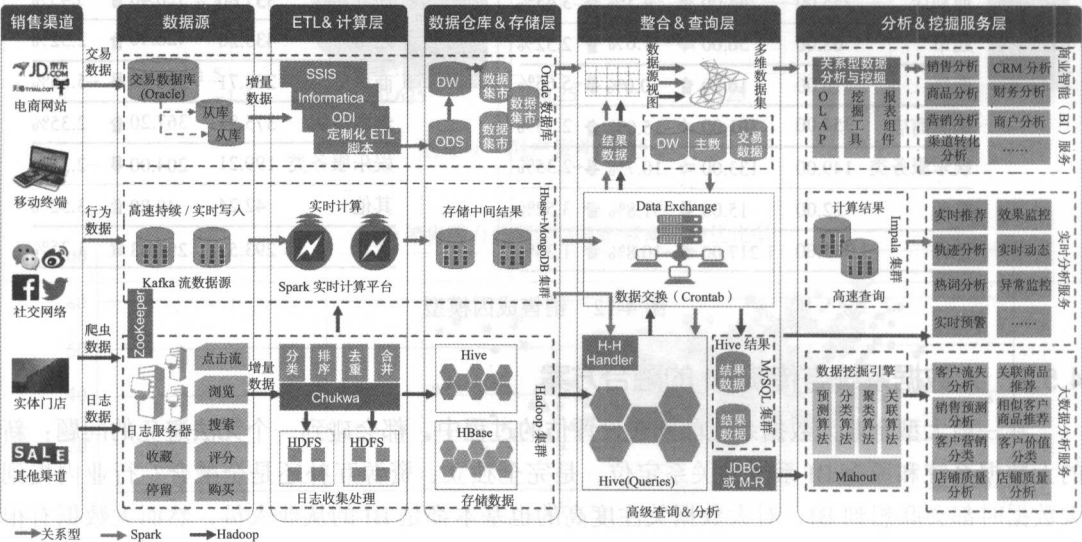


图 4-14 整体整合架构图

1. 数据采集部分

数据采集是大数据的基础,也是该企业大数据应用和提供服务的基础。数据采集可以分为以下几个类别:

- 自身数据:通过交易数据库接口、App 的开放 API、内嵌工具以及企业内部各系统所得到的商品、咨询信息、用户行为和反馈。这部分一般都是企业大数据采集的核心和骨架。
- 互联网公开数据:一部分是从各类门户、垂直网站、购买客户数据等通过网络爬虫获取的对商品、咨询、会员的侧面评价,是对核心数据的补充。另一部分通过 IP 数据、网络数据、商圈数据、天气数据等开放 API 接口获取,为数据的组织和分析提供多维度分析的支持。
- 合作企业数据:从企业级合作伙伴处获取的消费者电话、地址、关键成交因素等信息,尤其是高端客户的信息在提供时需要注重隐私和保密的数据。
- IT 系统及服务器运行数据:应用具体使用的物理资源,数据库和中间件资源;业务系统本身的监控状况,流量信息、存储增长信息、单据增长等。
- 其他数据:主数据和核心动态共享数据表、数据表关系,数据当前的分布和一致性、数据的生命周期、数据的增长趋势等。

2. 技术平台部分

应用场景包含以下几个方面:

原有业务系统及 BI 系统使用 Oracle 数据库存储、分析、挖掘功能不变,通过 DataIn (自主开发的数据迁移和同步组件) 组件并行实时同步存储到 Hadoop 中。互联网网页数据使用网络爬虫技术抓取数据,通过网页解析程序和中文分词处理,入库到 HDFS 文件中,并使用 Lucense\ Solr 建立全文索引。日志文件通过 Flume 组件采集,Chukwa 进一步处理后进行分发,离线部分存入 HDFS,实时部分推送到 Spark 进行实时关联计算。实时分析服务结果数据存入 Impala,离线结果数据存入 Hive。

3. 应用场景部分

应用场景包含以下几个方面:

- 会员模型:通过采集行为信息、注册信息、浏览器 Cookie 信息、历史交易信息、日常行为信息等进行模糊匹配、关键词匹配等构建用户画像,最直接的目的是精准定位,配合精准营销。
- 推荐引擎:根据用户的个人资料和行为数据匹配及推荐用户、产品和服务。在社交频道和交易频道用该方法为网上消费者推荐相关产品,同时为客户推荐相关的产品。
- 转化分析:通过对各门店、各频道访客行为数据的采集和分析,综合各个频道的目标、度量、KPI、维度和细分以决定店铺、商品、网站布局是否符合需求,确定布局、导航、优化路径。
- 社交分析:通过挖掘社交网络数据,一是确定社交网络中哪些会员对其他会员产生最大的影响力,有助于企业确定其“最重要”的会员(即最能够影响他人购买行为的会

员)；二是在宏观或微观层面确定用户对特定公司、品牌或产品的情绪。

- 监控平台：通过对大数据进行分析来获得可视化的业务和 IT 洞察力，该分析偏实时和准实时，重点是流程、数据和网络的实时管控，例如交易频道促销流量监控、重点客户行为实时分析等。
- 商务智能：通过全量数据的采集和分布式计算以及大数据的可视化技术，展现实时的运营分析。

4. 产品系统部分

产品系统方面主要包含以下内容：

- 数据魔方：基于海量的行业生态圈数据分析的商业数据产品，可以分析整个生态圈浏览、交易、收藏、搜索、生产、库存、物流等数据，结合用户特征，用于数据化运营、风险管控等。
- 精准营销：通过数据的整合，对用户进行清理、细分并标签化，在此基础上增加用户的价值，提升用户在平台上的黏性，最终促进营销的规模增长。从用户生命周期的角度分析用户管理体系，主要包括新用户获得、用户保持与提升、沉默用户唤醒和流失用户挽回等。
- 个性化推荐：根据大数据推荐模型运算，结合内部推荐渠道及外部推荐渠道，将消费者的需求推荐到网站、App、客服渠道、媒介、短信、邮件等所有用户有可能接收到的信息载体中，包括新用户引入、网站栏目推荐、个性画像推荐、商品推荐四大维度，全方位包含推荐算法的所有场景。
- 竞争情报：开放给合作伙伴，是设计团队、商户经营的战地局势图，记录每一天每一个商户的成长，解释其在对手中的位置，披露同行业竞争变化与商机。
- 其他：一类是利用所采集的大量数据，根据不同的组合向不同的会员提供个性化服务。另一类是面向内部管理者，提供各类在线、离线数据服务接口，支持业务方面的应用。

4.6 制定人才规划

企业的快速发展离不开人才队伍的规划和建设，在大数据人才稀缺的今天尤为如此。

4.6.1 指导思想

人才规划的出发点是全面实施人才强企战略，以加快提升人力资源职业能力为目标，紧紧抓住培养、吸引、用好人才三个环节，缓解人才短缺的状况，为公司发展提供人才支持，形成人才对企业发展的长效驱动力。

4.6.2 规划原则

以人为本的原则。要以实现人的可持续发展为目标，强化人才理念，促进人才队伍建设

与企业发展相协调,避免为了短期利益而损害人才发展和规划。

能力为本的原则。逐步形成以能力定岗位和职位的企业人力资源文化,加强人才建设的能力引导和信息传递,避免“元老”文化及“裙带”关系对能力的侵蚀。

终身培养的原则。推进实施多层次、多样化的企业内终身教育和培训,将企业短期应用培训与长期职业规划教育相结合,实现不同时期的供需平衡,并在人才结构方面实现协调和优化。

职业能力的原则。企业的人才规划和培养的目标是满足企业发展需求,因此需要优先进行满足岗位运营和工作需求的职业培训,形成人才培养+人才应用的双效模式。

整体推进的原则。加强对人才的培养、评价、选拔、使用、流动、激励、保障的衔接和相互作用,建立符合企业实际并能引领行业的优秀、高效运行机制和高质量工作模式。

4.6.3 核心内容

1. 人才培养

依托高校

依托高校指的是企业的大数据人才主要依托高校来培养。基于高校培养大数据人才主要包括以下几个方面:

- 依托高校定向培养大数据专业人才。
- 依托现有高校大数据课程,选拔优秀人才并输送到高校,促进关键人才的再教育。
- 大数据企业与高等院校、科研机构、职业培训等机构合作建立教育、实践基地,对大数据技术研发、市场推广、服务咨询等方面的人才进行岗位培训和职业教育。
- 支持高校、职业院校开展的大数据项目研究,并安排优秀人员一起参与相应实践。

自主培养

自主培养指的是企业通过内部资源自主培养大数据专业人才。自主培养的途径主要包括以下几个方面:

- 企业培养。通过公司内部的大数据专业化培训体系,建立完善的人才发展和培训机制,定期对专业课程、专业项目、专业技能等进行培训,以提高人才素质和能力。
- 自学成才。企业通过积极创造条件,提供发展平台,鼓励职工自学成才,在实践中成才。

2. 人才引进

在现有大数据人才稀缺的背景下,直接引入大企业的大数据人才或海外人才已经成为快速建立大数据体系的重要方式。人才引进的来源主要包括以下几个方面:

- 国内、外大型企业。目前大多数企业都直接从现有大公司挖掘专业技术人才。
- 国家科研机关、单位和相关机构。比如,中国科学院、工程院院士以及国内外学术、科学技术带头人和核心骨干。
- 著名高校。从著名高校中挖掘科研和教学带头人或学术精英加入到企业。

在人才引进时,通常包括个人引进和团队引进两种方式:

- 个人引进。个人引进指引入的是个体人才,通常对于核心骨干、业务带头人更加适用。

□ 团队引进。团队引进指引入的是一个工作团队，其中可能包含完成某项工作的各个核心岗位人才，通常对于新业务对象、中层执行层面的人才更加合适。

然而，在人才引进过程中，可能会面临一些问题：

□ 薪酬冲突。空降的人才薪资待遇往往大幅度高于公司内部培养的人才，这会导致同一公司内两种薪酬方案。对于这种情况，通常公司的做法是公开这种差异性，同时量化不同薪酬带来的市场回报。基于预期和实际回报来确定薪酬方案，而非单纯的职位、工龄、背景、资历等。

□ 文化冲突。空降的人才本身的工作环境 with 现有企业的工作环境通常具有很大差异，职位越高这点冲突往往会越明显。而且空降的人才由于没有跟随企业一同发展，对于企业内部工作文化的适应需要很长时间。

□ 人员流失。由于各种原因，空降的人才要真正发挥作用，需要在完全融入现有公司的前提下进行，如果没有完全融入团队，那么带来的是绩效、目标、成果的缺失，对空降人才造成负面影响，最终可能导致人才的流失。不仅如此，空降人才也可能在优厚待遇、技术能力、行业背景、资历经验、工作方法等方面对原有企业人才带来一定影响甚至排斥，因而也会导致原有公司人才的流失。

3. 创新孵化

面对信息化潮流，只有积极抢占制高点，才能赢得发展先机。在此背景下，“大众创业、万众创新”作为新常态下我国经济发展的一个重要引擎，也会加剧企业间的竞争态势。面对层出不穷的创业创新机会以及可能带来的市场风险，企业需要在内部提供此类创新孵化机制，这具有极其重要的战略意义：

第一，为企业发展提供新想法、新市场和新业态方向，利于完善现有企业的市场覆盖度并拓展新的市场机会。

第二，可以避免企业做大做强之后的大企业效率低下的问题，利于提高企业敏感度、信息传递效率、问题应对效率及执行反馈效率。

第三，源于企业内部创新孵化属于企业资源，即使出现与现有业务相冲突或具有替代关系的新业务体，也是内部市场和资产的主动转移，而非由于外部的竞争或直接市场吞并造成的企业危机，这是一种风险规避。

第四，内部孵化的与现有企业经营业务互补的创业主体可以形成集群规模，一方面可降低产业链或传统企业间的资源和信息流通、交换成本；另一方面也可以从现有企业的“单打独斗”走向“集群作战”，这也是增强现代企业在行业生态中竞争力的重要方法。

4. 人才激励

对于大数据企业内的人才激励，主要包含以下几种方案：

□ 目标激励。通过推行目标责任制，使企业经济指标层层落实，每个员工既有目标又有压力，从而产生强烈的动力，努力完成任务。

- 竞争激励。提倡企业内部员工之间、部门之间的有序平等竞争以及优胜劣汰。
- 荣誉激励。对员工劳动态度和贡献予以荣誉奖励,例如会议表彰、颁发荣誉证书、光荣榜、在公司内外媒体上的宣传报道、家访慰问、游览观光、疗养、外出培训进修、推荐获取社会荣誉、评选星级标兵等。
- 关怀激励。对员工工作和生活的关心,例如建立员工生日情况表、总经理签发员工生日贺卡、关心员工的困难和慰问或赠送小礼物。
- 物质激励。对于工作表现良好的员工增加期权、股份、分红、工资、生活福利、保险,发放奖金、奖励住房、生活用品、工资晋级;对于员工犯有过失、错误,违反企业规章制度,贻误工作,损坏设备设施,给企业造成经济损失和败坏企业声誉的员工或部门,分别给予警告、经济处罚、降职降级、撤职、留用察看、辞退、开除等处罚。

4.7 投入产出评估

在激烈竞争的经济环境下,成本控制和效益产出是企业普遍关注的焦点问题。如何科学地分析企业内部数据主体的成本构成及效益产出并找到最佳投入产出关系是每个数据管理者都在思考的问题。

作为企业内部的特殊主体,数据业务在某种程度上既是裁判员又是运动员,其效果投入和产出评估具有特殊性。

4.7.1 数据投入与产出的内涵

数据的投入与产出管理即数据的成本与效益管理。成本和效益分析作为一种评估和决策方法,将投入和产出关系分析运用于企业的计划决策之中,以寻求在投资决策上如何以最小的成本获得最大的收益。

这是一种量入为出的经营理念,要求对数据未来业务行动有较为明确的预期目标,并对完成预期目标有较大的信心。但是,大多数企业都没有做数据投入和产出管理,这导致企业对数据的预期目标和价值不明确、资源协调不及时甚至缺乏资源支持,数据价值无法发挥最大化。

数据投入与产出管理的基本前提是企业追求数据价值最大化。只有存在较为明确的产出预期,数据才能得到企业的重视和投入。良好的数据投入和产出管理能解答领导层在决策时的疑虑:

- 数据投入需要多少费用?
- 一次性投入费用和后期持续性费用预计是多少?
- 数据在什么时间能产出价值?
- 预期数据能产出什么价值?
- 这些价值能应用到企业的哪些环节,带来哪些提升?
- 运作起来的数据业务是否安全,存在哪些数据风险问题?



提示 很多时候企业领导层不是看不到数据的价值，而是对数据的投入和产出没有把握，因此会抱着试一试的想法让数据部门先运作起来，待数据产出价值之后再增加投入，这是一种常见的量入为出的管理思想。

4.7.2 数据投入与产出的特征

1. 逐利性

任何商业类企业都以营利为目的，任何管理和决策的目的都是利润最大化。数据投入与产出管理是企业成本收益分析的基本环节，在提高数据本身效益的基础上促进企业其他运营环节的业绩增长，甚至带来新的业绩增长点。

2. 最优化

数据投入产出管理的最佳状态是以最小的数据投入获得最大的数据产出，即所谓的 ROI（投入产出比）最大化。最优化是企业经营的理想状态。

3. 复杂性

（1）数据风险的不可避免性

数据投入和产出管理自身带有不可避免的风险性，不存在没有数据风险的企业，只是不同的企业风险大小有所差异而已。一方面，当企业不做投入和产出管理时，即使数据不被公司的任何业务部门使用，数据也会存在泄露、丢失和入侵风险；另一方面，当数据投入和产出管理工作时，可能由于流程、制度、人员等问题导致数据安全风险或数据决策风险问题。因此，凡是涉及数据的工作其风险性毫无疑问都会存在。

（2）数据的产出效果难以衡量

数据业务是依附于企业其他业务而存在的主体，如果没有业务部门或 IT 部门的辅助，数据将无法运转更不用说单独发挥价值。当对数据主体进行效果和产出评估时，由于存在这种依赖关系而无法准确识别效果提升的主要驱动因素，尤其是当业务主体、内部客观环境、外部企业环境等客观因素发生变化时，更无法进行效果评估。比如：

- ❑ 如果业务方在实施数据建议的同时也在优化其他因素，那么很难判断到底是业务的自身改善还是数据建议带来了效果提升。
- ❑ 如果因为企业外部因素如消费群体喜好变更、市场低迷等负面因素导致业务落地时效果不明显甚至出现负面效果，则很难准确找到原因主体。



提示 数据产出效果的不确定性及度量的复杂性是影响领导层对数据进行决策投入的最主要因素之一。站在企业领导层的角度思考：如果没有数据辅助决策，企业会损失多少利润？这个问题往往没有准确答案，并且也不可能通过数据方法（如 A/B 测试）得到答案。

4. 阶段性

当数据没有到达一定规模和量级时,很难通过数据规律的挖掘来大幅度提升业务效果,主要原因是数据量级不足以达到提炼规律的标准。只有当数据规模较大时,才存在提炼出规律和知识的可能性。换句话说,当企业较小时,数据可以发挥价值的场景以及产出效果不明显;当企业具备一定规模时,数据才有更好的环境发挥价值。因此,数据的作用在大多数情况下是锦上添花,而非雪中送炭。



提示 在企业成长初期,数据部门或相关职位通常不是必须的。即使在没有数据支持的情况下,如果企业具备其他核心竞争力(例如产品或服务)也会迅速做大。这说明了数据不是必须存在的主体,最起码在企业初期不是;只有企业在其他方面较为完善之后才会考虑数据问题。这时数据投入与产出评估的必要性才会显现出来。

4.7.3 数据投入与产出的管理

1. 数据投入管理

数据投入的组成

数据投入指开展数据工作需要的投入,包括固定投入费用、运营维护费用、时间成本、风险成本、机会成本以及数据获取成本。

(1) 固定投入费用

固定投入费用指开展数据工作初期需要投入的费用,通常是一次性固定费用,例如:

- 土地取得费用:自建数据中心时通常需要有自已的土地,因此会存在土地买卖或租赁费用。
- 机房建设费用:自建数据中心机房的费用,属于工程性的投入。
- 设备购置费用:数据中心正常工作所需的服务器、交换机、空调、UPS、备用发电机等设备的采购费用。
- 基础设施费用:数据中心内的道路、场地、供水、供电、供气、通信、排污、照明等费用。
- 系统和软件授权费用:数据处理时所需要的各种系统和软件授权费用,其中可能包含直接买断的费用投入,例如按 License 方式购买的软件。

(2) 运营维护费用

运营维护费用指为了保证数据工作正常开展所需的日常费用。

- 日常运维费用:包括水、电、气、通信、网络的维修、监测、保险等费用。
- 机房租赁成本:租赁第三方机房以及维护成本,购买虚拟主机或云服务的项目还存在服务租赁费用。
- 系统和软件授权费用:按照时间、规模、设备数等购买的年度或季节性费用。
- 人力成本:开展数据工作所需要的技术开发、产品设计、分析师、工程师等人员成本。

（3）时间成本

时间成本作为隐性成本通常被大多数企业忽视，但实际上在数据工作中如果投入管理不当，会导致大量时间消耗在不必要的环节，例如复杂的数据清洗、内部无效沟通、重复项目建设等，而对真正产出价值的数据挖掘上投入不足。

（4）风险成本

风险成本包括数据泄露风险及数据应用风险可能带来的企业损失。数据泄露意味着第三方可能利用企业自身数据开展针对性的经营措施，而数据应用风险将可能带来数据决策失误，这两种风险都可能对企业形成难以估量的损失。

（5）机会成本

机会成本指在决策过程中面临多个选择时，放弃的可能产出最大价值的选择，也叫作“替代性成本”，通俗的理解就是“有得必有失”。机会成本对于数据投入与产出管理的意义在于：当企业集中大量资源到数据主体时，如果把这些资源聚焦到产品研发、服务提升、流程规范上可能带来更高的企业效益。因此，存在投资到企业其他业务主体上产生更大价值回报的可能性；同样，当数据部门面对多个项目工作时，在资源有限的情况下需要选择投入和产出效果最好的项目。

（6）数据获取成本

数据按照来源可分为企业采集、数据购买、数据交换以及其他途径，在这些来源中，数据购买需要直接投入成本，而数据交换也是通过数据作为等价物换取的一种方式，这些都是数据获取成本的体现。

数据投入的评估

数据投入的评估最重要的三个指标是：总成本、平均成本以及边际成本。

（1）总成本

总成本是所有成本之和，公式为：

$$\text{总成本} = \text{固定投入费用} + \text{运营维护费用} + \text{数据获取成本} + \text{其他成本}$$

在成本计算中，隐性成本如时间成本、风险成本、机会成本等无法具体量化，因为无法体现在公式中，这部分可通过文字的形式进行定义描述，也可结合现有数据价值进行适当的定量评估。另外，企业内部成本主体之间的变更（费用摊派到哪个部门）也会影响数据总成本。

（2）平均成本

平均成本根据不同的平均主体可延伸出不同的指标，例如平均用工成本。平均成本是衡量每个单位投入的重要指标。



做企业数据成本评估时，通常对于长期使用的经营性资产采用按照其使用年限每年分摊购置成本的会计处理方法。比如，企业购置了价值 500 万元的数据工具，如果按照 10 年进行摊销，每年摊销成本为 50 万元。这 50 万元将在不同年限对相应部门做

成本核算时单独计算，而不是在购买初期就将全部成本纳入成本计算范畴。摊销常见于土地使用权、大型软件、开办费等无形资产，而对于固定资产则采用折旧的处理方法。

（3）边际成本

在经济学和金融学中，边际成本指每新增一个单位产品（生产或销售产品）带来的总成本的增量。在衡量数据主体过程中，定义为每新增一个单位产品（生产或销售产品）带来的数据总成本的增量。

众所周知，数据工作在前期需要投入一定的固定费用而导致数据单位成本较高，如图 4-15 中的阶段一所示；随着业务数据量的增加以及数据规模化效应的显现，每增加一次销售所需要的单位数据成本会逐渐降低并达到相对稳定的水平，如图 4-15 中的阶段二所示。

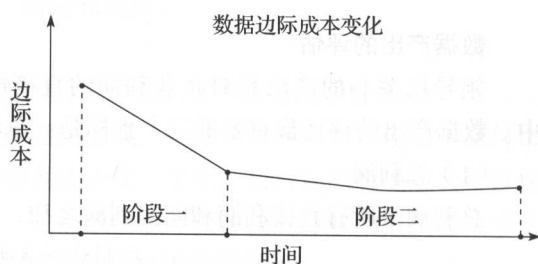


图 4-15 数据边际成本变化

2. 数据产出管理

数据产出的组成

数据产出指数据最终对企业利润和价值的贡献，而不是数据报表、产品、模型或报告。数据产出主要包括业务价值提升及数据现金流贡献。

（1）业务价值提升

数据产出的最初预期是通过数据对业务运营起到效果提升作用，常见的价值产出场景包括：

- ❑ 降低营销成本并提高营销 ROI；
- ❑ 提高站内用户体验并最终提高用户订单转化率；
- ❑ 增加客户黏性并提高客户重复购买率；
- ❑ 降低库存积压率并减少资金占用，提高库存周转率。
- ❑ 业务价值提升是辅助决策的主要目标，需要借助业务部门落地动作才能实现价值，通过数据优化前和优化后的效果对比来评估提升效果。

（2）数据现金流贡献

数据现金流贡献指直接通过数据带来的新业务模式或新的销售增长点，例如：

- ❑ 通过数据发现了新的广告位，从而带来新的广告售卖利润点；
- ❑ 通过对用户行为的精准匹配，在网站内部开发出个性化推荐系统，通过对内部商家开放并获取佣金、返点、服务费等形式增加企业收入点；
- ❑ 通过用户需求分析发现了新的市场方向，并通过新模式的开拓增加企业收入来源；
- ❑ 某些情况下，数据还可以作为一项单独的业务体变现，例如直接售卖数据或通过自有

数据为第三方数据提供清洗和校验等服务。

数据现金流直接将数据作为业务主体，通过 IT 系统或工具的辅助支撑实现利润提升；在为业务部门提供新的业务模式时，也可以直接给业务带来从无到有的价值增长。数据现金流贡献必须基于数据驱动的机制才能实现。



提示 数据作为企业的核心资产之一正被越来越多的企业关注，未来数据质量度、数据完整度、数据量级、数据类型多样性、数据新鲜度等都可能被标准化并成为企业资产负债表中的重要项目，这是数据作为隐性资产的重要特性。

数据产出的评估

领导层关心的产出是对企业利润的直接贡献，该贡献可以反映到企业财报或会计报表中。数据产出的评估最重要的三个指标是：总利润、平均利润以及边际利润。

(1) 总利润

总利润是所有直接利润和间接利润之和，公式为：

$$\text{总利润} = \text{间接利润} + \text{直接利润}$$

间接利润即业务提升价值，计算方式为：

$$\text{间接价值} = \text{数据优化后产出} - \text{数据优化前产出}$$

直接业务价值即数据现金流，直接通过数据产出而非业务建议或动作产出的利润贡献。

(2) 平均利润

平均利润根据不同的平均主体有不同的指标，例如人均利润等。

(3) 边际利润

在经济学中，边际利润或边际收益指每新增一个单位产品（生产或销售产品）带来的总利润或收益的增量。在衡量数据主体过程中，该定义为每新增一个单位产品（生产或销售产品）带来的数据总利润的增量。

如图 4-16 所示为数据边际利润变化趋势：阶段一，企业数据发展初期，边际利润伴随着数据工作规模的扩大而逐渐上升并体现出规模效应；阶段二，在企业发展到一定规模时，数据部门的边际利润出现瓶颈，即使增加更多的资源业务也无法获得更高的边际利润；阶段三，当企业规模非常大时，如果不控制数据资源的投入，会由于资源浪费、人浮于事等导致边际利润的下降。

3. 数据投入与产出优化

数据投入与产出优化不仅仅是单纯的压缩成本和费用，它需要结合企业战略目标、经营方向、经营模式等建立科学合理的成本分析与产出评估体系，从投入和产出两方面入手进行控制。

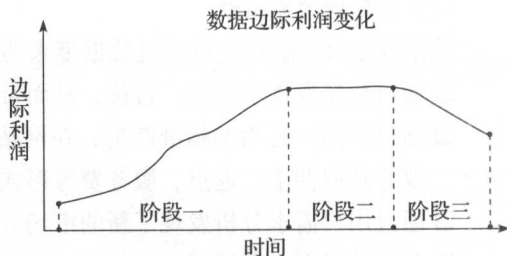


图 4-16 数据边际利润变化

作为数据管理者的主要工作已经不是凡事身体力行、事必躬亲，而是要有效地领导和管理他人做事。数据投入与产出优化，主要从项目管理（管事）、人员管理（管人）和流程管理（管流程）三方面入手。

数据项目管理

项目是在一定条件及限定资源下需要完成的任务。数据类项目根据难易程度、完成周期、涉及范围等进行区分，大型项目可能包括数据仓库建设、BI 产品开发、站内个性化系统开发等，小型项目可能是一个数据分析报告、一次数据挖掘模型、一次调研报告等。

项目管理的本质是确定要做什么、做到什么程度、什么时候做、谁来做、如何做等。数据类项目管理的五要素为：范围、时间、成本、质量和风险。

（1）范围管理

范围管理是对数据工作涉及的内容、程度及产出的定义，其中包括两层含义：

- 定义工作产出目标，即如何衡量项目工作已经完成，例如交付客户流失模型、交付市场调研报告、交付报表系统或者站内推荐系统上线。工作产出目标的明确定义可以保证项目工作者准确掌握项目的结束状态，并能减少额外资源投入导致的浪费以及带来的负面影响，例如项目无法正常交付、影响其他项目正常开展等。
- 定义工作需求条件，即要完成该项目需要哪些条件和资源的支持，主要包括数据范围和人员范围。数据范围包括数据主题、数据类型、数据格式、抽取条件、时间范围、数据粒度等；人员范围指哪些人参与才能实现项目需求。

（2）时间管理

时间管理是为了确保按照项目预定时间完成的一系列管理过程，包括活动排序、时间估计、进度安排及时间控制等各项工作。

时间管理通常使用项目进度表来描述，进度表不仅能说明完成项目工作所需的时间，也规定了每个活动的具体开始和完成日期，甚至可以对每个活动的主要内容及预期目标进行简单定义。图 4-17 所示为一个简单的产品项目时间进度表。

ID	任务名称	开始时间	完成时间	持续时间	2014年09月													
					8	9	10	11	12	13	14	15	16	17	18	19		
1	需求分析	2014/9/8星期一	2014/9/8星期一	1d														
2	需求策划	2014/9/9星期二	2014/9/10星期三	2d														
3	产品开发	2014/9/11星期四	2014/9/15星期一	3d														
4	产品测试	2014/9/16星期二	2014/9/18星期四	3d														
5	产品上线	2014/9/19星期五	2014/9/19星期五	1d														

图 4-17 产品项目时间进度表

（3）成本管理

成本管理是项目管理的重要内容，每个项目工作必须在既定的预算内完成。通常成本指

的是显性成本，例如咨询费用、系统购买付费、培训费用、设备资金等，某些情况下也可以评估隐性成本因素，尤其是对项目完成影响较大的客观因素及不可量化因素。在范围管理中主要体现产出什么，而在成本管理中主要体现投入什么，通过范围管理和成本管理形成两端的约束，配合其他管理要素共同实现过程管理。

（4）质量管理

质量管理是为了确保项目达到预定目标对质量进行要求和规范的管理过程，包括质量规划、质量控制和质量标准等。质量管理对于工程类和 IT 类项目较为容易量化，例如数据仓库建设项目可通过数据延迟性、数据完整度、ETL 时间、数据仓库性能测试等进行管理；但是对于业务类项目，例如报告、模型等很难使用标准化指标进行衡量，对于这类项目通常通过两方面进行质量管理：

一是过程管理，通过对过程的规范来保证过程实施的标准化和规范化。

二是结果管理，通过对项目结果的基本结构、模块、撰写标准、用户理解性、用户满意度甚至对业务直接驱动产生的价值进行评估。

（5）风险管理

数据类项目与其他项目管理的不同点在于其风险管理要求较高，这一方面体现在数据安全性对于企业战略安全的意义，另一方面也体现在数据的应用风险和决策失误风险管理的必要性。关于数据风险的相关知识，请见 4.8 节。

数据人员管理

人员管理包括目标管理、成长管理、绩效管理。

（1）目标管理

目标管理分为三个层次：

□ 价值认同感。团队成员必须认同企业和数据部门的文化及价值观，不认同企业文化和价值观的员工不仅自身无法带来价值还会带坏团队的风气并影响绩效产出。

□ 为现在而工作。每个团队成员都必须了解自己工作岗位的基本职责、内容和基本产出，这是保证现有工作正常进行并实现既定目标产出的基础。

□ 为未来而工作。为每个团队成员设计未来发展方向和职业前景，驱动团队成员发挥超出预期的工作能量并带来更好的工作产出。

（2）成长管理

团队成长是管理者必须重视的课题，一个没有成长的团队一定会被企业和社会淘汰。企业内部的团队成长主要通过两条路径：

一是团队自我成长。这个过程中需要管理者提供尽量多的工作场景和实践机会，同时能够信任团队并适度放权，在工作方向正确的前提下允许一定程度的犯错。除此以外，适时地引入新的血液，新方法、新思路和新见解对团队激励有好处，鲶鱼效应会让团队更有动力。

二是部门内部培训。数据部门需要根据工作重点、内容和目标建立内部培训计划，针对内部数据工作人员进行培训，内容包括业务类知识、数据类知识、工具类知识甚至流程类知

识, 这些知识可以避免团队走弯路, 并提高工作绩效产出。

(3) 绩效管理

绩效管理是驱动人员产出的制度保障, 目的是持续提升个人、部门和组织的绩效产出。绩效管理并不只是绩效考核, 而是包括管理者和员工为了达到目标而共同参与的绩效制定、绩效沟通、绩效考核和绩效反馈提升的循环过程。

- 绩效制定: 根据企业和部门目标制定绩效指标, 兼具业绩、态度和能力三方面。
- 绩效沟通: 绩效标准与团队成员的沟通, 保证团队成员准确理解目标内容并为之努力。
- 绩效考核: 定期评估团队成员实际表现, 并得出考核结果。
- 绩效反馈提升: 与团队成员沟通考核结果并明确改进方向。



提示 绩效管理并不等于扣钱, 也不是无条件的升职加薪, 而是根据评估结果驱动成长或优胜劣汰。作为数据部门的 Leader, 维护并扩大团队的工作资源和团队成员利益是基本职责之一。

数据流程管理

数据流程管理指的是流程分析、流程定义、资源分配、时间安排、流程质量、流程参与及规范、流程优化的过程。常见的数据流程包括需求处理流程、数据产品开发流程、项目工作流程、数据落地流程、数据培训流程等。有关数据流程的具体内容见 2.3 节。

4.8 数据风险管理

4.8.1 数据风险管理的概念

数据风险指企业在使用数据(采集、存储、挖掘和应用等)过程中所面临的风险, 其产生原因通常是企业内部数据架构、流程、制度和标准不完善, 数据不仅不能为企业带来促进作用, 反而存在危害企业发展的可能。

为此, 数据风险管理应运而生。数据风险管理指对企业内数据使用过程中的风险进行识别、评估、分析和处理, 通过一定的措施进行风险管控, 以实现数据安全并有效应用的管理方法。

数据资产作为企业资产的一部分, 正越来越成为企业核心竞争力的主要组成部分。数据风险管理也与政策风险管理、战略风险管理、财务风险管理、市场风险管理、运营风险管理、团队风险管理和法律风险管理等共同组成企业风险管理体系。因此, 数据风险管理与控制也将成为企业风险管理的重要组成部分。

数据风险管理不是单独存在的一个实体, 而是贯穿于整个企业内部各个环节的管理机制, 它具有以下几个特点:

- 它的目的是协助实现企业和部门的目标。

- 它的实现依附于其他业务主体，涉及企业内部数据流转相关的所有中心、部门和人员，贯穿于业务过程的始终。
- 它是辅助于战略和战术目标实现的一种方法，但不是唯一方法。
- 它的内涵包括目标确定、风险识别与评估和风险处理与监督三层，而不仅仅是风险处理。

4.8.2 数据风险管理的类型

数据风险管理与控制涉及数据整个工作流程，包括数据采集、数据存储、数据分析挖掘和数据应用，但从风险管理的方向来看，可以分为两类：数据信息安全管理 and 数据应用安全管理。

1. 数据信息安全管理

数据信息安全管理是在数据工作过程中为防止信息泄露、信息丢失、数据损坏等信息危害的管理措施，管理对象是数据本身。

数据信息泄露管理

数据信息泄露管理主要指对企业内部的营销规划、客户信息、财务数据、商品数据、库存数据等关乎企业核心竞争力的机密资料和数据进行安全管理，以防止被其他主体或个人获取。如果这些数据被竞争对手窃取并利用，将会对企业生产经营造成不可估量的灾难性影响。

信息泄露可能发生在数据从产生到最终应用的各个环节，主要产生场景如下：

- 内部人员的无意泄密。由于系统中毒而导致的信息资料被窃取、无意间将企业内部信息携带到外部环境、与外部人员沟通时无意识的数据披露等都会导致泄密情况的发生。
- 内部人员的有意泄密。内部人员出于利益、权利和其他关系，有意出卖公司内部数据而导致信息泄露，这是信息安全管理的应对场景和监督对象。
- 数据查看权限失控。企业内部的不同级别人员的权限管理不当，造成可访问数据范围、导出和保存权限的混乱等，这些将增加高级机密或核心数据被获取并传播的风险。
- 加密管理问题。企业内、外部流转的数据文档、文件等，通常需要根据不同的应用场景和应用权限进行加密，一旦加密管理制度缺乏良好的实施管理，将导致信息传播时不受控制。
- 黑客或间谍活动。很多黑客和间谍通过技术手段入侵企业内部的数据节点以获得关键信息，这是企业数据安全中最重要，也是最为困难的部分。



提示 数据信息泄露的人员对象不仅包括企业外部人员，还包括企业内部人员。凡是被不具有知晓权利的相关人员获得了数据信息都属于数据信息泄露的范畴。

数据信息丢失管理

数据信息丢失管理指通过一定措施防止数据丢失的管理方法，数据丢失通常发生在三个阶段：

- ❑ 数据采集丢失。在数据采集阶段数据就已经发生丢失的情况，常见于在数据开发和采集阶段由于软件、硬件或客观情况的变更导致的数据采集缺失。比如，服务器故障、硬盘空间不足、软件编译执行问题、数据采集规划不全面、业务需求变更等都会导致这个问题。



注意

随着业务客观环境、运营水平以及需求的变化，业务对数据需求可能发生改变；而数据开发则不可能立即根据业务需求调整数据采集策略，这就要求在数据采集初期，数据分析师需要与业务、产品经理、开发工程师详细沟通数据需求，并在开发初期提供尽量完善的数据监测和采集方案，避免后期进行大规模的数据二次开发。

- ❑ 数据存储丢失。在数据存储过程中，由于存储设备的逻辑原因如感染病毒、误格式化、设备断电、文件系统错误等，物理原因如硬盘损坏，固件原因如固件使用次数过多导致的无法正常识别数据设备等导致的数据丢失，另外地震、洪水、雷电等不可抗力因素也可能导致数据存储丢失。
- ❑ 数据编辑丢失。对于数据的删除或清空操作如数据删除、表删除、表清空等权限，必须进行严格的权限限制，否则将可能导致严重的数据丢失问题。

数据信息质量管理

数据信息质量管理指对数据质量进行把控，通过维护或提高数据质量来保证后期数据应用的稳定性及可靠性，良好的数据质量是产生价值的基本前提。数据信息质量的下降通常由三个原因导致：

- ❑ 数据采集验证缺陷。在数据采集初期，如果没有良好的数据质量验证和数据校验规则，可能出现所采集的数据质量度低，表现为数据空值多、数据长度短并丢失重要信息、数据缺乏关联主键、数据键值匹配错误等，数据采集缺陷很难通过后期数据维护进行修复。因此，在数据采集阶段的数据质量验证是保证数据质量的第一步。
- ❑ 数据编辑权限失控。对于数据可编辑操作如更新、插入等，需要区分场景和对象进行权限控制，否则可能由于操作问题而影响数据质量，最终导致“垃圾数据”的产生。



提示

在大多数情况下，数据的任何操作都是在非生产数据库中进行，这意味着即使出现数据误操作也能通过生产数据库重新更新；但即使这样，数据权限的严格管理仍然必不可少，重新更新所消耗的服务器资源、时间成本、人力成本以及对其他关联业务的影响都构成潜在的风险，某些情况下可能是重大影响，例如实时数据决策、到期的应付账款等。

- ❑ 数据生命周期原因。很多企业在应用数据过程中，不注意数据生命周期的管理和维护，因此导致数据质量度的下降及后期数据应用问题的产生。

提示 任何事物都有生命周期，数据也不例外。数据生命周期包括在线阶段（包括采集、处理、应用等）、离线阶段（主要指数据备份和归档）、销毁阶段（数据删除或销毁），定义数据的生命周期不仅能保证数据质量度，还能提高数据应用效率和效果，同时可减轻对 IT 运维的负担。关于数据生命周期的话题会在 8.6.3 节“数据生命周期管理”部分详细介绍。

2. 数据应用风险管理

数据应用风险管理指企业在应用数据过程中通过一定流程、机制和标准，指导数据应用的可理解性、可应用性以及有效性，最大化减少数据应用风险的管理机制。

提示 目前大多数企业的数据风险管理主要集中在信息安全管理，而对于数据应用风险管理关注较少。数据应用风险管理是数据辅助决策、驱动业务落地的有效手段，同时也是盘活数据资产的重要保障。但由于数据应用风险管理过程中需要大量兼具业务知识、数据知识的复合型人才，且应用过程中对于管理目标的制定尚没有统一标准，因此建立并完善这一内容需要足够的知识、经验、耐性和信心。

数据应用风险常出现在数据抽样、数据处理、数据分析和挖掘、数据展现、数据理解 and 数据应用这六个场景。

- ❑ 数据抽样错误，指数据样本选择时出现的错误，主要表现为使用错误的抽样方法、选择过少的样本量，这些问题最终导致数据样本无法代表全部样本特征或无法满足分析需求，而基于此分析产生的结论也将缺乏可信度。
- ❑ 数据处理错误，包括对数据异常值、空值、错误值的处理错误，数据抽取格式、条件和转换错误等，这些将导致原始数据的质量度偏低，甚至某些异常值还会对后期的分析和挖掘工作造成直接影响。
- ❑ 数据分析和挖掘错误，主要指在数据分析和挖掘过程中使用了错误的分析和挖掘方法或选择了拟合程度较低的模型等，最终将导致结论的错误或不可信。比如，使用变量间的相关关系来代替因果关系做数据预测。
- ❑ 数据展现错误，由于图形扭曲、特殊坐标轴处理、特殊标记、数据单位转换等造成的数据展现错误，直接结果是导致数据汇报对象的理解错误。
- ❑ 数据理解错误。由于业务方自身知识、经验和能力造成的数据理解偏差，可能导致后期在数据应用和落地时出现误导。

□ 数据应用错误。比如，将 Apriori 模型结果应用于具有明显序列特征的关联分析场景中，使用时间序列预测具有自变量的预测场景，将变量在小范围的拟合关系扩大到大范围场景中应用等。

数据应用对企业或业务部门最终产生的影响可分为三类：第一类是正面的影响，即从数据中获得了更好的决策建议；第二类是无影响，即数据没有产生任何作用；第三类是消息的影响，即企业或业务部门被数据误导，从而制定了错误的决策。上述六个场景在大多数情况下会带来第三类影响，而数据应用风险管理的核心针对第三类影响。



注意 在企业内部，数据相关部门是一个介于业务体系、职能体系与技术体系之间的交叉部门，因此其架构可能出现在上述三个体系中。但作为不直接产生实际业务动作或业务价值的部门，如果对企业发展产生负面影响，那么就需要考虑数据部门存在的意义。因此，数据是一把双刃剑，能否用好关键看自己。

但是，上述六个场景也只是表象特征，我们由表及里进行分析，到底又是哪些根本性原因产生了这些表象？笔者在此归纳为两个方面：

一是数据本身的局限性，包括数据采集初期样本量不足、数据非结构化特征、数据分析和挖掘算法的局限性、数据只能反映结果但无法解释原因等，这些因素导致了数据在处理问题时会遇到无法跨越的障碍，同时由于其无法进行自我鉴定和评估，更无法实现自我修复。比如，抽样不能保证小范围数据每次都被抽中的概率，基于历史环境的数据结论在面对新环境时会显得力不从心。

二是数据应用流程的不规范性。数据在业务应用之前需要经过多个步骤，无论哪个步骤出现问题都会对后期应用产生影响。如果这个过程缺乏必要的培训、监督、约束等条件，那么每个环节都会由于不可控而导致数据应用结果不理想。

4.8.3 数据风险管理的原则

1. 树立正确的数据安全观念

众所周知，数据是企业的重要资产之一，运用得当时还会成为企业的核心竞争力；但如果因为惧怕数据风险而将数据限制起来，那么数据将无法实现其价值，甚至企业还要为维护数据而花费额外的硬件、软件、人力和时间成本等，最终它将成为企业的“负资产”。

作为企业的领导者和数据部门的负责人，需要平衡数据风险与回报的关系，在降低数据风险的基础上产生最大的数据回报，而任何只看风险或只看回报的做法都是不当的。

同时，作为企业内部相关人员，在接触数据的整个过程中都需要树立风险意识、增强防范措施，避免由于自身导致的数据风险问题，数据安全永远是任何数据动作的首要前提。

2. 建立系统的数据安全机制

数据安全机制并不是某个部门或某个人员的事情，数据安全机制需要从数据源头抓起，

并跟踪到数据落地应用的最终环节，因此数据安全涉及所有与数据有关的业务部门；同时，除了数据自身的流程规范之外，还需要有相关管理部门进行风险把控和监督，例如计划部、管理部、内审部等。综合防范和治理是建立安全机制的基础。

除了制度和流程约束外，数据安全机制的落实还需要有自动化、智能化的工作方式，专门的技术或工具必不可少，例如流程跟踪系统、权限控制系统、预警工作系统、防入侵系统等。

4.8.4 数据风险管理与控制

数据风险管理与控制的对象包括业务主体和数据主体两部分；从数据风险管理与控制的流程来看，分为目标设定、识别与评估、处理与监督三个阶段。

1. 数据风险目标设定

数据风险目标设定包括业务目标设定和数据目标设定。

业务目标设定

业务目标基于企业和业务应用主体设定，通常将 KPI 设定为业务目标，示例如下：

- 销售额提升 50%。
- 妥投率提升 25%。
- 会员数增加 20000。

这些目标的设定随着业务主体的不同而有所差异，而同一业务主体在不同时间内的目标也会发生变化。因此，业务目标设定值变化较为频繁，属于动态目标范畴。

数据目标设定

与业务目标相对的是数据目标，数据目标是针对数据主体产生，目的是保证数据的最终产出价值，并对数据主体的各个方面、不同环节进行评估。由于变化幅度小、调整频率低，数据目标属于静态目标范畴，具体包括以下几个方面：

(1) 数据质量目标

数据质量通常从完整性、一致性、及时性、有效性、正确性、真实性六个方面进行衡量。

- 完整性：数据的完整程度，指没有丢失数据记录或数据值的比例。
- 一致性：同一维度或分析对象在不同库、表间的标识相同、约束条件一致、数据值相等。
- 及时性：数据更新是否能满足业务应用需求程度，具体视业务需求而定，如果业务需要每小时应用数据，数据及时性要求就是小时；如果是每天看数据，及时性要求就是天。
- 有效性：数据是否符合应用需求以及满足需求的程度。
- 正确性：数据正确与否以及正确的比例。
- 真实性：评估数据的来源是否真实，识别其中的造假数据。

(2) 数据安全目标

数据安全评估通常从以下几个方面入手。

- 备份覆盖度：指进行备份的数据覆盖程度，同时基于不同数据的重要性还可设置备份频率、备份份数、备份介质数等评估维度。

- 防护覆盖度：指入侵检测系统、防火墙、杀毒软件、密保措施、权限控制、身份验证措施对所有服务器、客户端的覆盖比例。
- 预警覆盖度：预警机制对所有数据范围的覆盖程度，无论是访问异常、数据异常等均可进行信息预警。
- 物理安全性：对数据机房物理环境、服务器环境进行进入控制、监控与警报的覆盖程度。
- 动态保护系数：通过动态准入地址、口令、密码、数字签名等进行的保护覆盖程度。

(3) 数据应用目标

数据应用目标主要用来评估数据在应用过程中的质量情况以及所产生的价值。

- 数据可接触比例：评估应该接触到数据但由于各种原因未能接触到数据的比例。
- 数据到达率：评估数据实际接触的业务主体与理论上应该接触到业务主体的比例。
- 数据覆盖范围：度量数据在业务应用各个环节的参与和覆盖程度。比如，通常情况下数据只在发生业务产生动作之后参与进来，但数据还可以在业务动作发生之前、发生过程中参与进来。
- 数据正确率：基于数据的业务驱动效果可通过数据做进一步的衡量，正确率即正确场景占有所有驱动场景的比例。
- 直接数据价值：通过数据自身的建议或落地动作直接提升的销售、订单和利润情况，排除业务改进带来的贡献，只评估数据的贡献。
- 间接数据价值：指通过数据建议以及业务自身优化共同产生提升的销售、订单和利润贡献情况。
- 数据自动化程度：评估数据工作体系和流程中系统自动化的实现程度，自动化程度越高可以减少更多的人力、物力和时间成本，从而提高数据应用效率和应用效果。
- 数据智能化程度：评估数据自我学习和智能工作的程度，数据智能化程度是数据产生价值的重要途径，同时也是未来数据工作发展的重要方向。

2. 数据风险识别与评估

数据风险识别与评估是针对数据工作过程中可能产生的潜在风险进行评估和判断，以确定风险产生的可能性以及存在条件和影响的过程。

数据风险识别主要依靠两个方面开展：一是数据部门与业务部门的主观经验（包含风险感知和经验判断）；二是客观数据（包含数据结论、预警模型等）。

数据风险识别和评估包括以下几方面内容：

- 数据质量风险：判断数据是否存在质量隐患，例如缺失、不一致、错误、造假、不及时等。
- 数据安全风险：判断数据是否存在安全隐患，重点是非法使用、丢失、入侵、损坏等。
- 应用风险评估：对于业务应用和实施过程中存在的数据误导及其他影响业务主体和数据主体目标实现的重大隐患进行评估。

数据风险识别的基本原则是：

- ❑ 周期性原则：数据风险识别与评估是一次性工作，而且需要定期开展和维护。
- ❑ 差异性原则：不同周期、不同频率内的识别范围、监测方法和实施过程是不同的。
- ❑ 系统性原则：数据风险识别与评估是一个系统性工作，需要全部相关部门和人员参与。
- ❑ 全面性原则：数据风险识别需要针对所有相关节点和环节开展，而不应该仅仅关注其中某几个环节，全面性原则要求对数据流通的所有环节进行监督。

数据风险评估的结果根据其影响的类型可分为三种：

- ❑ 负面影响：风险结果是负面的，对企业和部门发展有消极影响。
- ❑ 无影响：风险结果没有任何影响。
- ❑ 正面影响：风险结果是正面的，对企业和部门发展有积极影响。



提示 在数据风险识别与评估过程中，应尽量将评估结果量化。除了“无影响”之外的正面影响和负面影响都需要指出影响类型、影响范围、影响频率、影响程度，理想情况下需要与公司核心业务和利润挂钩，如此评估结果才能更有说服力；除了显性影响外，其他隐性影响也可以作为评估参考结果，例如时间成本、机会成本等。

3. 数据风险处理与监督

数据风险的处理措施包括以下四种：

- ❑ 规避：通过一定措施规避数据风险，包括权限的关闭、多数据节点的备份和冗余、数据文件永久性加密、数据信息销毁、寻找替代性数据解决方案等。
- ❑ 减轻：通过一定措施减轻数据风险带来的负面影响，在规避中的措施同样适用于减轻处理。
- ❑ 接受：不采取任何措施，接受数据风险带来的所有影响，包括积极影响和消极影响。
- ❑ 扭转：通过一定方式扭转数据带来的负面影响，并使之产生积极或正面意义。比如，通过数据风险问题而衍生出新的业务模式或盈利方式。

以上的处理结果通常是基于当次风险事件而产生的策略调整或补救措施，同时企业内部还需要建立或完善数据风险管理机制和流程体系，并通过持续监督落实进行必要调整。如图 4-18 所示为数据风险管理与控制体系的主要内容，这些



图 4-18 数据风险管理与控制体系

内容在本章的前面都已经涉及。

为了保证数据风险管理与控制体系的有效落地，需要一定的机制进行保证，如图 4-19 所示。

- 第一级企业监管部门，包括计划部、管理部、内审部和 HR 部门等，这些部门通过对流程、事件和关键节点的管理与审查，从机制上保证数据风险管理的制度落地；其中的 HR 部门还会通过 KPI 考核、数据风险管理得分等机制直接与相关人员的绩效挂钩。
- 第二级工具监督控制，企业内部的加密工具、权限配置工具、验证工具、防护工具、识别工具等是建立自动化实时监控、多重保护、动态管理和不同安全层级的基础，同时也是预防的主要措施。数据权限最小化、针对不同主体的分层管理、隔离信息主体与访问主体是基础原则。
- 第三级数据部门监控，包括数据备份、流程跟踪、数据质量管理、完善数据流程和指标，通过定期检查、预警机制和数据的智能应用等提高数据风险管理水平，对员工的数据风险管理培训也是必不可少的内容。
- 第四级人员自我约束，人是工作的主体，解决了人的问题即可解决大多数数据风险问题。除了需要内部员工提高数据安全意识、法律意识、流程意识和标准意识外，还需要同步提高其应用技能，包括安全防护技能、工具使用技能、数据工作技能以及其他相关技能，这些技能是“由知到行”的必要条件。



图 4-19 数据风险管理与控制落地



提示 在数据风险处理与监督过程中，不仅要产生风险的直接主体进行管理，还要对产生风险的根源进行有效防治，这样才能达到“标本兼治”的效果。

4.9 本章小结

企业大数据实施之前的规划是企业大数据之路的必然步骤，本章从整体上梳理了大数据规划时涉及的各个方面并结合案例阐述了如何进行方案落地。

本章需要读者重点掌握的知识点包括：

- 掌握整个规划的基本环节和思路，并能实际应用到企业规划中；
- 制定目标蓝图和建设目标的基本方法和思路，想清楚未来要做什么才能有进一步如何去做的的问题；
- 掌握技术方案中选型和基本方案的设计方法，并能跟企业现有 BI 体系融合起来；
- 掌握数据投放产出的基本定义和内涵，并能根据企业实际进行数据资产和数据业务的价值评估。

大数据技术介绍

提到大数据技术经常会提到 Hadoop，简单回顾一下 Hadoop 的前世今生：Hadoop 原是基于 Nutch 项目中的一个组件，用做爬虫 URL 和网页数据的存储，能更有效、廉价地存储和计算海量数据。在 Nutch0.8.0 版本之前，Hadoop 是内置在 Nutch 中的一部分。从 Nutch0.8.0 开始，NDFS 和 MapReduce 从中被剥离出来成立一个新的开源项目 Hadoop，而 Nutch0.8.0 版本的架构比从前有了根本性的变化，完全构建在 Hadoop 的基础之上了，鉴于其最初的设计思路和架构，Hadoop 在海量数据处理方面固然有它独特的优势。本章将详细讲解大数据的核心技术、相关技术以及算法库。

5.1 核心技术

5.1.1 Hadoop 生态

Hadoop 是采用分布式架构用于大规模数据存储和计算的系统，是 Apache 基金会的开源项目，起源于谷歌的三大论文：GFS、Map/Reduce、Bigtable，由 Doug Cutting 发起，实现在大量计算机组成的集群中对海量数据进行分布式计算。Hadoop 框架中最核心设计就是：MapReduce 和 HDFS，除此之外还有一系列基于该组件之上的相关组件。其中，MapReduce 提供了对数据的计算，HDFS 提供了海量数据的基本存储功能。

有这样的 Hadoop 发起者，把高深莫测的搜索技术形成产品，贡献给普罗大众；还是他，打造了目前在云计算和大数据领域里如日中天的 Hadoop。还有无数个默默奉献的开源者的共同努力，才有了 Hadoop 现在的备受瞩目，叩开了大数据时代的海量数据存储与处理的大门，形成了企业级海量数据存储与计算及其应用的新领地。

1. HDFS

HDFS 是 Hadoop 的分布式文件系统，用于实现大规模数据可靠的分布式读写。HDFS 针对的使用场景是数据读写具有“一次写，多次读”的特征，而数据“写”操作是顺序写，也就是在文件创建时的写入或者在现有文件之后的添加操作。HDFS 保证一个文件在一个时刻只被一个调用者执行写操作，而可以被多个调用者执行读操作。整个过程如图 5-1 所示。

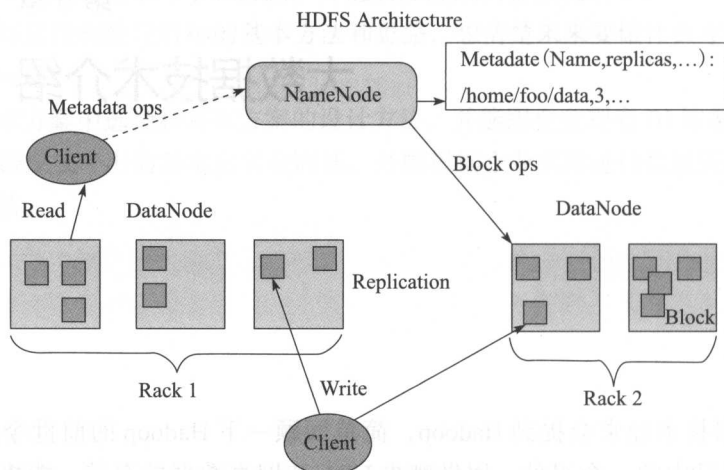


图 5-1 分布式文件系统 HDFS 工作流程

图 5-1 中展现了整个 HDFS 的三个重要角色：NameNode、DataNode 和 Client。NameNode 可以看做是分布式文件系统管理者，主要负责管理文件系统的命名空间、集群配置信息和存储块的复制等。NameNode 会将文件系统的 Metadata 存储在内存中，这些信息主要包括文件信息、每一个文件对应的文件块的信息和每一个文件块在 DataNode 的信息等。DataNode 是文件存储的基本单元，它将 Block 存储在本地文件系统中，保存了 Block 的 Metadata，同时周期性地将所有存在的 Block 信息发送给 NameNode。Client 就是需要获取分布式文件系统文件的应用程序。

这里通过三个操作来说明它们之间的交互关系。

文件写入：

- Client 向 NameNode 发起文件写入的请求。
- NameNode 根据文件大小和文件块配置情况，返回给 Client 所管理部分 DataNode 的信息。
- Client 将文件划分为多个 Block，根据 DataNode 的地址信息，按顺序写入到每一个 DataNode 块中。

文件读取：

- Client 向 NameNode 发起文件读取的请求。
- NameNode 返回文件存储的 DataNode 的信息。

□ Client 读取文件信息。

文件 Block 复制:

□ NameNode 发现部分文件的 Block 不符合最小复制数或者部分 DataNode 失效。

□ 通知 DataNode 相互复制 Block。

□ DataNode 开始直接相互复制。

HDFS 特点

作为大数据 Hadoop 底层分布式存储系统, HDFS 提供了非常值得借鉴学习的分布式存储解决方案:

□ Block 的放置: 默认不配置。一个 Block 会有三份备份, 一份放在 NameNode 指定的 DataNode, 另一份放在与指定 DataNode 非同一 Rack 上的 DataNode, 最后一份放在与指定 DataNode 同一 Rack 上的 DataNode。备份无非就是为了数据安全, 考虑同一 Rack 的失败情况以及不同 Rack 之间数据拷贝性能问题就采用这种配置方式。

□ 心跳检测 DataNode 的健康状况, 如果发现问题就采取数据备份的方式来保证数据的安全性。

□ 数据复制 (场景为 DataNode 失败、需要平衡 DataNode 的存储利用率和需要平衡 DataNode 数据交互压力等情况): 这里先说一下, 使用 HDFS 的 balancer 命令, 可以配置一个 Threshold 来平衡每一个 DataNode 磁盘利用率。比如, 设置了 Threshold 为 10%, 那么执行 balancer 命令时, 首先统计所有 DataNode 的磁盘利用率的均值, 然后判断如果某一个 DataNode 的磁盘利用率超过这个均值 Threshold, 那么将会把这个 DataNode 的 Block 转移到磁盘利用率低的 DataNode, 这对于新节点的加入来说十分有用。

□ 数据校验: 采用 CRC32 作数据校验。在文件 Block 写入时除了写入数据还会写入校验信息, 在读取时需要校验后再读入。

□ HA 解决方案: 即 Active Master 不断将信息写入一个共享存储系统, 而 Standby Master 则不断读取这些信息, 以与 Active Master 的内存信息保持同步。当需要主备切换时, 选中的 Standby Master 需先保证信息完全同步后, 再将自己的角色切换至 Active Master。目前而言, 常用的共享存储系统有以下几个: Zookeeper、NFS、HDFS、Bookkeeper 和 QJM。

□ HDFS Federation: 它让多个 NameNode 分管不同的目录进而实现访问隔离和横向扩展。对于运行中 NameNode 的单点故障, 通过 NameNode 热备方案 (NameNode HA) 实现。

□ 数据管道性的写入: 当客户端要写入文件到 DataNode 上, 首先客户端读取一个 Block 然后写到第一个 DataNode 上, 然后由第一个 DataNode 传递到备份的 DataNode 上, 一直到所有需要写入这个 Block 的 DataNode 都成功写入, 客户端才会继续开始写下一个 Block。

□ 安全模式: 在分布式文件系统启动时, 开始时会有安全模式, 当分布式文件系统处于安全模式的情况下, 文件系统的内容不允许修改也不允许删除, 直到安全模式结

束。安全模式主要是为了系统启动时检查各个 DataNode 上数据块的有效性，同时根据策略必要地复制或者删除部分数据块。运行期通过命令也可以进入安全模式。在实践过程中，系统启动时去修改和删除文件也会有安全模式不允许修改的出错提示，只需要等待一会儿即可。

HDFS 优点

在系统架构方面，有以下优点：

(1) 容错性

容错性是所有分布式的系统集群都需要面临的重要问题，由于集群规模的扩大，对于硬件故障将成为常态，而不是异常。整个 HDFS 系统将由数百或数千个存储着文件数据片断的服务器组成。实际上它里面有非常巨大的组成部分，每一个组成部分都很可能出现故障，这就意味着 HDFS 里总是有一些部件是失效的，因此，故障的检测和自动快速恢复是 HDFS 一个很核心的设计目标。

(2) 高吞吐

运行在 HDFS 之上的应用程序必须流式地访问它们的数据集，它不是运行在普通文件系统之上的普通程序。HDFS 被设计成适合批量处理的，而不是用户交互式的。重点是在数据吞吐量，而不是数据访问的反应时间，POSIX 的很多硬性需求对于 HDFS 应用都是非必须的，去掉 POSIX 一小部分关键语义可以获得更好的数据吞吐率。

(3) 扩展性

HDFS 被设计用来部署在低廉的硬件上，可支持硬件的水平添加。对于 hadoop1.x 来说由于组件元数据的限制，单集群的最大规模被限制在 4000 ~ 5000 台。hadoop2.x 版本以后，通过对新增 YARN、NameNode 等设计的改造，摆脱了这个限制。

2. MapReduce

MapReduce 是 Hadoop 的核心，是 Google 提出的一个软件架构，用于大规模数据集（大于 1TB）的并行运算。概念“Map”（映射）和“Reduce”（化简）及它们的主要思想，都是从函数式编程语言借来的，还有从矢量编程语言借来的特性。

它极大地方便了编程人员在不会分布式并行编程的情况下，将自己的程序运行在分布式系统上。当前的软件实现是指定一个 Map 函数，用来把一组键值对映射成一组新的键值对，指定并发的 Reduce 函数，用来保证所有映射的键值对中的每一个共享相同的键组。图 5-2 为 MapReduce 的执行过程。

图 5-2 是论文里给出的流程图。一切都是从最上方的 User Program 开始的，User Program 链接了 MapReduce 库，实现了最基本的 Map 函数和 Reduce 函数。图中执行的顺序都用数字标记了。

第一步：MapReduce 库先把 User Program 的输入文件划分为 M 份（ M 为用户定义），每一份通常有 16MB ~ 64MB，如图左方所示分成了 $\text{split } 0 \sim 4$ ；然后使用 fork 将用户进程拷贝到集群内其他机器上。

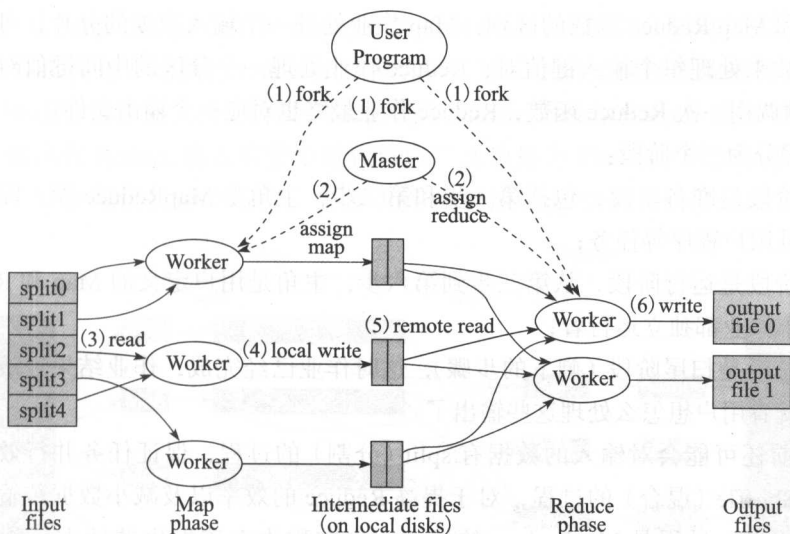


图 5-2 MapReduce 工作流程图

第二步：User Program 的副本中有一个称为 Master，其余称为 Worker，Master 是负责调度的，为空闲 Worker 分配作业（Map 作业或者 Reduce 作业），Worker 的数量也是可以由用户指定的。

第三步：被分配了 Map 作业的 Worker，开始读取对应分片的输入数据，Map 作业的数量是由 M 决定的，和 split 一一对应；Map 作业从输入数据中抽取出键值对，每一个键值对都作为参数传递给 Map 函数，Map 函数产生的中间键值对被缓存在内存中。

第四步：缓存的中间键值对会被定期写入本地磁盘，而且被分为 R 个区， R 的大小是由用户定义的，将来每个区会对应一个 Reduce 作业；这些中间键值对的位置会被通报给 Master，Master 负责将信息转发给 Reduce Worker。

第五步：Master 通知分配了 Reduce 作业的 Worker 它负责的分区在什么位置（肯定不止一个地方，每个 Map 作业产生的中间键值对都可能映射到 R 个不同分区），当 Reduce Worker 把所有它负责的中间键值对都读过来后，先对它们进行排序，使得相同键的键值对聚集在一起。因为不同的键可能会映射到同一个分区也就是同一个 Reduce 作业（谁让分区少呢），所以排序是必须的。

第六步：Reduce Worker 遍历排序后的中间键值对，对于每个唯一的键，都将键与关联的值传递给 Reduce 函数，Reduce 函数产生的输出会添加到这个分区的输出文件中。

第七步：当所有的 Map 和 Reduce 作业都完成了，Master 唤醒正版的 User Program，MapReduce 函数调用返回 User Program 的代码。

所有执行完毕后，MapReduce 输出放在了 R 个分区的输出文件中（分别对应一个 Reduce 作业）。用户通常并不需要合并这 R 个文件，而是将其作为输入交给另一个 MapReduce 程序处理。整个过程中，输入数据是来自底层分布式文件系统（HDFS）的，中间数据是放在本地文件系统的，最终输出数据是写入底层分布式文件系统（HDFS）的。而且我们要注意 Map/

Reduce 作业和 Map/Reduce 函数的区别：Map 作业处理一个输入数据的分片，可能需要调用多次 Map 函数来处理每个输入键值对；Reduce 作业处理一个分区的中间键值对，期间要对每个不同的键调用一次 Reduce 函数，Reduce 作业最终也对应一个输出文件。

上述流程分为三个阶段：

- ❑ 第一阶段是准备阶段，包括第一步和第二步，主角是 MapReduce 库，完成拆分作业和拷贝用户程序等任务；
- ❑ 第二阶段是运行阶段，从第三步到第六步，主角是用户定义的 Map 和 Reduce 函数，每个小作业都独立运行着；
- ❑ 第三阶段是扫尾阶段（剩下的步骤），这时作业已经完成，作业结果被放在输出文件中，就看用户想怎么处理这些输出了。

在 Map 前还可能会对输入的数据有 split（分割）的过程，保证任务并行效率，在 Map 之后还会有 Shuffle（混合）的过程，对于提高 Reduce 的效率以及减小数据传输的压力有很大的帮助。Shuffle 过程是 MapReduce 的核心，也被称为奇迹发生的地方。所以要理解 MapReduce，Shuffle 是必须要了解的。

Shuffle 的本义是洗牌、混洗，把一组有一定规则的数据尽量转换成一组无规则的数据，越随机越好。MapReduce 中的 Shuffle 更像是洗牌的逆过程，把一组无规则的数据尽量转换成一组具有一定规则的数据。为什么 MapReduce 计算模型需要 Shuffle 过程？我们都知道 MapReduce 计算模型一般包括两个重要的阶段：Map 是映射，负责数据的过滤分发；Reduce 是规约，负责数据的计算归并。Reduce 的数据来源于 Map，Map 的输出即是 Reduce 的输入，Reduce 需要通过 Shuffle 来获取数据。

可能大家更熟悉的是 Java API 中的 Collections.shuffle（List）方法，它会随机地打乱参数 list 中的元素顺序。如果你不知道 MapReduce 中的 Shuffle 是什么，那么请看图 5-3。

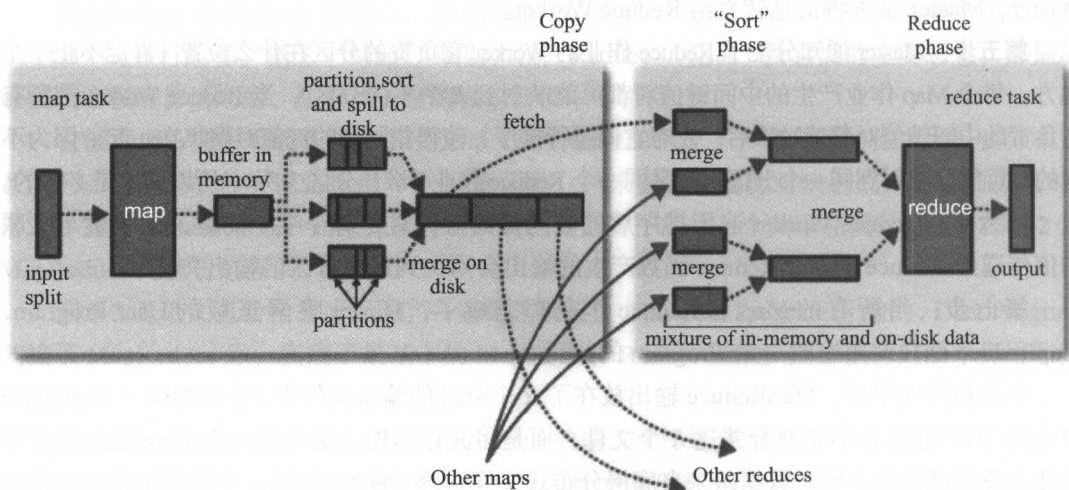


图 5-3 Shuffle 过程中的 Map 分解图

实际上,从 Map Task 任务中的 `map()` 方法中的最后一步调用即输出中间数据开始,一直到 Reduce Task 任务开始执行 `reduce()` 方法结束,这一中间处理过程就被称为 MapReduce 的 Shuffle。Shuffle 过程分为两个阶段:Map 端的 Shuffle 阶段和 Reduce 端的 Shuffle 阶段。

从 Map 输出到 Reduce 输入的整个过程可以广义地称为 Shuffle。Shuffle 横跨 Map 端和 Reduce 端,在 Map 端包括 spill 过程,在 Reduce 端包括 copy 和 sort 过程,如图 5-4 所示。

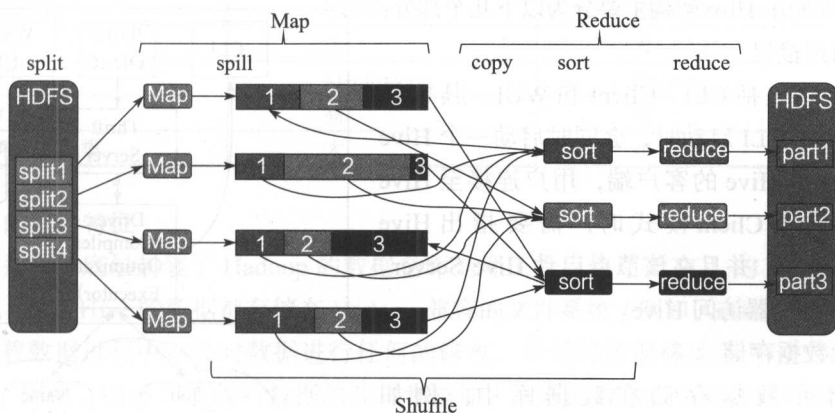


图 5-4 Shuffle 过程中的 Reduce 分解图

Spill 过程包括输出、排序、溢写、合并等步骤,执行流程:

- 利用文件分割器将大文件分割成多个 Part。
- 每个 part 小文件分配给一个 Map 任务,并对每个小文件中的数据执行 Map 函数。
- Map 任务执行完成后将中间结果输出到内存中进行 Shuffle 与排序操作。
- sort 排序操作执行完成后将结果输出到 Reduce 任务中进行 reduce 操作。
- Reduce 函数执行完成后将结果输出到 HDFS 中。

MapReduce 框架的优点:

- 大规模数据处理:隐藏并行细节、简化开发工作。
- 自动并行化:Map 和 Reduce 函数高度并行。
- 伸缩性好:资源扩展简单,几何线性扩展。

3. Hive

Hive 是基于 Hadoop 的一个数据仓库工具,可以将结构化的数据文件映射为一张数据库表,并提供简单的 SQL 查询功能,可以将 SQL 语句转换为 MapReduce 任务进行运行。其优点是学习成本低,可以通过类 SQL 语句快速实现简单的 MapReduce 统计,不必开发专门的 MapReduce 应用,由于 SQL 的通用性使得 Hive 十分适合数据仓库的统计分析。

如上所述,Hive 是建立在 Hadoop 上的数据仓库基础构架。它提供了一系列的工具,可以用来进行数据提取转化加载(ETL),这是一种可以存储、查询和分析存储在 Hadoop 中的大规模数据的机制。Hive 定义了简单的类 SQL 查询语言,称为 HQL,它允许熟悉 SQL 的

用户查询数据。同时，这个语言也允许熟悉 MapReduce 的开发者开发自定义的 mapper 和 reducer 来处理内建的 mapper 和 reducer 无法完成的复杂的分析工作。

Hive 功能架构

Hive 没有专门的数据格式，用户可以对数据格式进行自定义，另外 Hive 可以很好地以接口的方式提供查询服务，服务底层采用 Thrift RPC 框架。功能逻辑如图 5-5 所示。

如图 5-5 所示 Hive 架构主要分为以下几个部分：

(1) 用户接口

用户接口包括 CLI、Client 和 WUI。其中最常用的是 CLI，CLI 启动时，会同时启动一个 Hive 副本。Client 是 Hive 的客户端，用户连接至 Hive Server。在启动 Client 模式时，需要指出 Hive Server 所在节点，并且在该节点启动 Hive Server。WUI 是通过浏览器访问 Hive。

(2) 元数据存储

Hive 将元数据存储在数据库中，例如 MySQL、Derby。Hive 中的元数据包括表的名字、表的列和分区及其属性、表的属性（是否为外部表等）、表的数据所在目录等。

(3) 解释器、编译器、优化器、执行器

解释器、编译器、优化器完成 HQL 查询语句从词法分析、语法分析、编译、优化到查询计划的生成。生成的查询计划存储在 HDFS 中，并在随后由 MapReduce 调用执行。

Hive 的数据存储在 HDFS 中，最新的 Hive 版本支持 Local MapReduce Job 和集群版的 MapReduce Job，通常数据量比较小时，Hive 计算引擎可以采用 Local MapReduce Job 提升计算效率，然而大数据量或者大部分的复杂查询均由集群版的 MapReduce 完成，另外包含 * 的查询，例如 `select * from tbl` 不会生成 MapReduce 任务。

Hive 应用场景

Hive 构建在基于静态批处理的 Hadoop 之上，Hadoop 通常都有较高的延迟并且在作业提交和调度时需要大量的开销。因此，Hive 并不能够在大规模数据集上实现低延迟快速的查询，例如 Hive 在几百 MB 的数据集上执行查询一般有分钟级的时间延迟。

Hive 并不适合那些需要低延迟的应用，例如联机事务处理（OLTP）。Hive 查询操作过程严格遵守 Hadoop MapReduce 的作业执行模型，Hive 将用户的 HiveQL 语句通过解释器转换为 MapReduce 作业提交到 Hadoop 集群上，Hadoop 监控作业执行过程，然后返回作业执行结果给用户。Hive 并非为联机事务处理而设计，Hive 并不提供实时的查询和基于行级的数

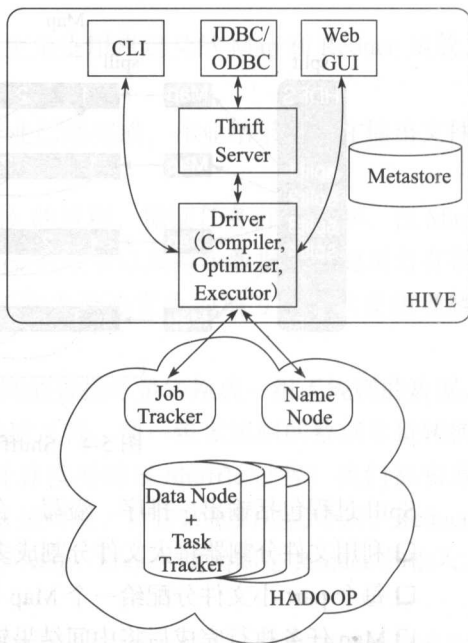


图 5-5 Hive 的功能架构示意图

据更新操作。Hive 的最佳使用场合是大数据集的批处理作业，例如网络日志分析。表 5-1 是 Hive 和普通关系型数据库的对比。

表 5-1 Hive 和普通关系型数据库的对比

	Hive	RDBMS
查询语言	HQL	SQL
数据存储	HDFS	Raw Device or Local FS
索引	无	有
执行	MapReduce	Excutor
执行延迟	高	低
处理数据规模	大	小

Hive 的设计特征

Hive 是一种底层封装了 Hadoop 的数据仓库处理工具，使用类 SQL 的 HiveQL 语言实现数据查询，所有 Hive 的数据都存储在 Hadoop 兼容的文件系统（例如 Amazon S3、HDFS）中。Hive 在加载数据过程中不会对数据进行任何的修改，只是将数据移动到 HDFS 中 Hive 设置的目录下，因此，Hive 不支持对数据的改写和添加，所有的数据都是在加载时确定的。Hive 的设计特点如下：

- ❑ 支持索引，加快数据查询。
- ❑ 不同的存储类型，例如纯文本文件、HBase 中的文件。
- ❑ 将元数据保存在关系数据库中，大大减少了在查询过程中执行语义检查的时间。
- ❑ 可以直接使用存储在 Hadoop 文件系统中的数据。
- ❑ 内置大量用户函数 UDF 来操作时间、字符串和其他的数据挖掘工具，支持用户扩展 UDF 函数来完成内置函数无法实现的操作。
- ❑ 类 SQL 的查询方式，将 SQL 查询转换为 MapReduce 的 job 在 Hadoop 集群上执行。

Hive 数据存储结构

首先，Hive 没有专门的数据存储格式，也没有为数据建立索引，用户可以非常自由地组织 Hive 中的表，只需要在创建表时告诉 Hive 数据中的列分隔符和行分隔符，Hive 就可以解析数据。

其次，Hive 中所有的数据都存储在 HDFS 中，Hive 中包含以下数据模型：表（Table）、外部表（External Table）、分区（Partition）、桶（Bucket）。

- ❑ Hive 中的 Table 和数据库中的 Table 在概念上是类似的，每一个 Table 在 Hive 中都有一个相应的目录存储数据。比如，一个表 pvs，它在 HDFS 中的路径为：/wh/pvs，其中，wh 是在 hive-site.xml 中由 \${hive.metastore.warehouse.dir} 指定的数据仓库的目录，所有的 Table 数据（不包括 External Table）都保存在这个目录中。

- ❑ Partition 对应于数据库中的 Partition 列的密集索引，但是 Hive 中 Partition 的组织方式和数据库中的很不相同。在 Hive 中，表中的一个 Partition 对应于表下的一个目

录，所有 Partition 的数据都存储在对应的目录中。比如，pvs 表中包含 ds 和 city 两个 Partition，则对应于 ds = 20090801, ctry = US 的 HDFS 子目录为：/wh/pvs/ds=20090801/ctry=US；对应于 ds = 20090801, ctry = CA 的 HDFS 子目录为：/wh/pvs/ds=20090801/ctry=CA。

❑ Buckets 对指定列计算 hash，根据 hash 值切分数据，目的是并行，每一个 Bucket 对应一个文件。将 user 列分散至 32 个 Bucket，首先对 user 列的值计算 hash，对应 hash 值为 0 的 HDFS 目录为：/wh/pvs/ds=20090801/ctry=US/part-00000；hash 值为 20 的 HDFS 目录为：/wh/pvs/ds=20090801/ctry=US/part-00020。

❑ External Table 指向已经在 HDFS 中存在的数 据，可以创建 Partition。它和 Table 在元数据的组织上是相同的，而实际数据的存储则 有较大的差异。Table 包括创建过程和数据加载过程（这两个过程可以在同一个语句中完成），在加载数据的过程中，实际数据会被移动到数据仓库目录中；之后的数据访问将会直接在数据仓库目录中完成。删除表时，表中的数据和元数据将会被同时删除。External Table 只有一个过程，加载数据和创建表同时完成（CREATE EXTERNAL TABLE……LOCATION），实际数据是存储在 LOCATION 后面指定的 HDFS 路径中，并不会移动到数据仓库目录中。当删除一个 External Table 时，仅删除元数据，表中的数据不会真正被删除。

Hive 在使用过程中的优化方法

❑ join 连接时的优化：当三个或多个以上的表进行 join 操作时，如果每个 on 使用相同的字段连接时只会产生一个 mapreduce。

❑ join 连接时的优化：当多个表进行查询时，从左到右表的大小顺序应该是从小到大。原因：Hive 在对每行记录操作时会把其他表先缓存起来，直到扫描最后的表进行计算。

❑ 在 where 字句中增加分区过滤器。

❑ 当可以使用 left semi join 语法时不要使用 inner join，前者效率更高。原因：对于左表中指定的一条记录，一旦在右表中找到立即停止扫描。

❑ 如果所有表中有一张表足够小，则可置于内存中，这样在和其他表进行连接时就能完成匹配，省略掉 reduce 过程。设置属性即可实现，set hive.auto.covert.join=true；用户可以配置希望被优化的小表的大小 set hive.mapjoin.smalltable.size=2500000；如果需要使 用这两个配置可置入 \$HOME/.hiverc 文件中。

❑ 同一种数据的多种处理：从一个数据源产生的多个数据聚合，无需每次聚合都重新扫描一次。

❑ limit 调优：limit 语句通常是执行整个语句后返回部分结果。set hive.limit.optimize.enable=true；。

❑ 开启并发执行：某个 job 任务中可能包含众多的阶段，其中某些阶段没有依赖关系可以并发执行，开启并发执行后 job 任务可以更快地完成。设置属性：set hive.exec.parallel=true；。

□ Hive 提供的严格模式，禁止 3 种情况下的查询模式。

当表为分区表，where 字句后没有分区字段和限制时，不允许执行；

当使用 order by 语句时，必须使用 limit 字段，因为 order by 只会产生一个 reduce 任务；
限制笛卡儿积的查询。

□ 合理的设置 map 和 reduce 数量。

□ jvm 重用。可在 Hadoop 的 mapred-site.xml 中设置 jvm 被重用的次数。

4. HBase

HBase (Hadoop Database) 是一种高可靠性、高性能、面向列、可伸缩的分布式存储系统，利用 HBase 技术可在廉价的 PC Server 上搭建起大规模结构化存储集群。

HBase 逻辑架构

HBase 是 Google Bigtable 的开源实现，类似 Google Bigtable 利用 GFS 作为其文件存储系统，HBase 利用 Hadoop HDFS 作为其文件存储系统；Google 运行 MapReduce 来处理 Bigtable 中的海量数据，HBase 同样利用 Hadoop MapReduce 来处理 HBase 中的海量数据；Google Bigtable 利用 Chubby 作为协同服务，HBase 利用 Zookeeper 作为对应。如图 5-6 所示为 HBase 的功能架构。

图 5-6 描述了 Hadoop EcoSystem 中的各层系统，其中 HBase 位于结构化存储层，Hadoop HDFS 为 HBase 提供了高可靠性的底层存储支持，Hadoop MapReduce 为 HBase 提供了高性能的计算能力，Zookeeper 为 HBase 提供了稳定服务和 failover 机制。

此外，Pig 和 Hive 还为 HBase 提供了高层语言支持，使得在 HBase 上进行数据统计处理变得非常简单。Sqoop 则为 HBase 提供了方便的 RDBMS 数据导入功能，使得传统数据库数据向 HBase 中迁移变得非常方便。

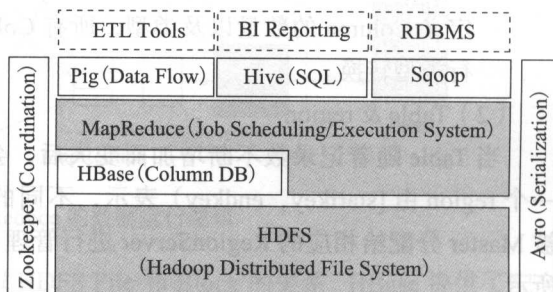


图 5-6 HBase 的功能架构示意图

HBase 操作访问接口

□ Native Java API：最常规和高效的访问方式，适合 Hadoop MapReduce Job 并行批处理 HBase 表数据

□ HBase Shell：HBase 的命令行工具，最简单的接口，适合 HBase 管理使用。

□ Thrift Gateway：利用 Thrift 序列化技术，支持 C++，PHP，Python 等多种语言，适合其他异构系统在线访问 HBase 表数据。

□ REST Gateway：支持 REST 风格的 Http API 访问 HBase，解除了语言限制。

□ Pig：可以使用 Pig Latin 流式编程语言来操作 HBase 中的数据，和 Hive 类似，本质最终也是编译成 MapReduce Job 来处理 HBase 表数据，适合做数据统计。

HBase 数据模型

(1) Table & Column Family

HBase 以表的形式存储数据。表由行和列簇组成，其数据模型如表 5-2 所示。

表 5-2 数据模型示意

Row Key	Timestamp	Column Family	
		URI	Parser
r1	t3	url=http://www.taobao.com	title= 天天特价
	t2	host=taobao.com	
	t1		
r2	t5	url=http://www.alibaba.com	content= 每天…
	t4	host=alibaba.com	

- ❑ Row Key: 行键, Table 的主键, Table 中的记录按照 Row Key 排序。
- ❑ Timestamp: 时间戳, 每次数据操作对应的时间戳, 可以看做是数据的 version number。
- ❑ Column Family: 列簇, Table 在水平方向由一个或者多个 Column Family 组成, 一个 Column Family 中可以由任意多个 Column 组成, 即 Column Family 支持动态扩展, 无需预先定义 Column 的数量以及类型, 所有 Column 均以二进制格式存储, 用户需要自行进行类型转换。

(2) Table & region

当 Table 随着记录数不断增加而变大后, 会逐渐分裂成多份 split, 成为 region (区域), 一个 region 由 [startkey, endkey) 表示, 不同的 region 会被 Master 分配给相应的 RegionServer 进行管理, 如图 5-7 所示。

(3) -ROOT- & .META. Table

HBase 中有两张特殊的 Table, -ROOT- 和 .META., 如图 5-8 所示。

- ❑ META.: 记录了用户表的 Region 信息, .META. 可以有多个 region。
- ❑ -ROOT-: 记录了 .META. 表的 Region 信息, -ROOT- 只有一个 region。
- ❑ Zookeeper 中记录了 -ROOT- 表的 location。

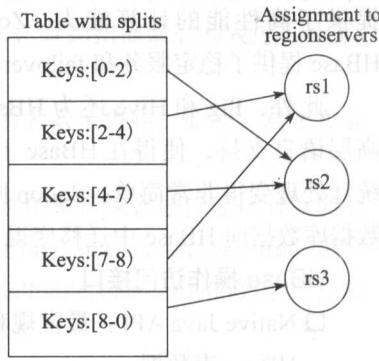


图 5-7 数据存储区域

Client 访问用户数据之前需要首先访问 Zookeeper, 然后访问 -ROOT- 表, 接着访问 .META. 表, 最后才能找到用户数据的位置去访问, 中间需要多次网络操作, 不过 Client 端会做 cache 缓存。

(4) MapReduce on HBase

在 HBase 系统上运行批处理运算, 最方便和实用的模型依然是 MapReduce, 如图 5-9 所示。

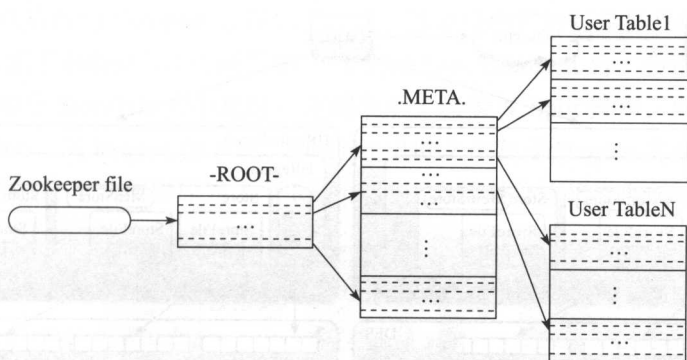


图 5-8 -ROOT- 和 META 表

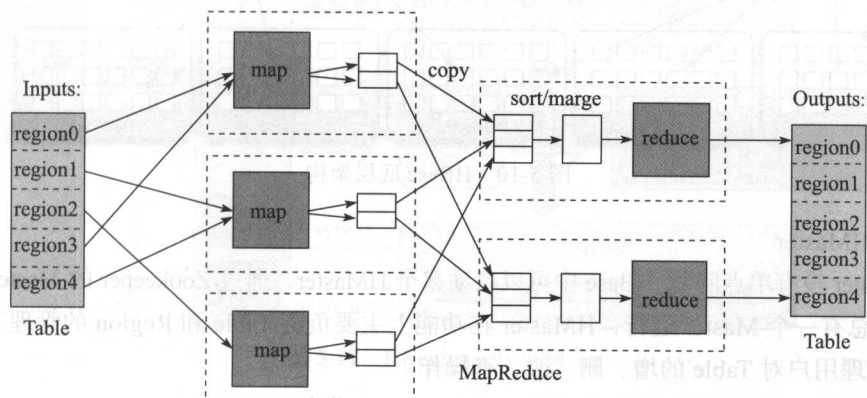


图 5-9 基于 MapReduce 的数据运行逻辑

HBase Table 和 Region 的关系，比较类似 HDFS File 和 Block 的关系，HBase 提供了配套的 TableInputFormat 和 TableOutputFormat API，可以方便地将 HBase Table 作为 Hadoop MapReduce 的 Source 和 Sink，对于 MapReduce Job 应用开发人员来说，基本不需要关注 HBase 系统自身的细节。

HBase 底层系统架构

HBase 是基于 Hadoop 之上的数据库应用，底层采用 HDFS 作为存储，具体的架构如图 5-10 所示。

(1) Client

HBase Client 使用 HBase 的 RPC 机制与 HMaster 和 HRegionServer 进行通信，对于管理类操作，Client 与 HMaster 进行 RPC；对于数据读写类操作，Client 与 HRegionServer 进行 RPC。

(2) Zookeeper

Zookeeper Quorum 中除了存储了 -ROOT- 表的地址和 HMaster 的地址，HRegionServer 也会把自己以 Ephemeral 方式注册到 Zookeeper 中，使得 HMaster 可以随时感知到各个 HRegionServer 的健康状态。此外，Zookeeper 也避免了 HMaster 的单点问题，见下文描述。

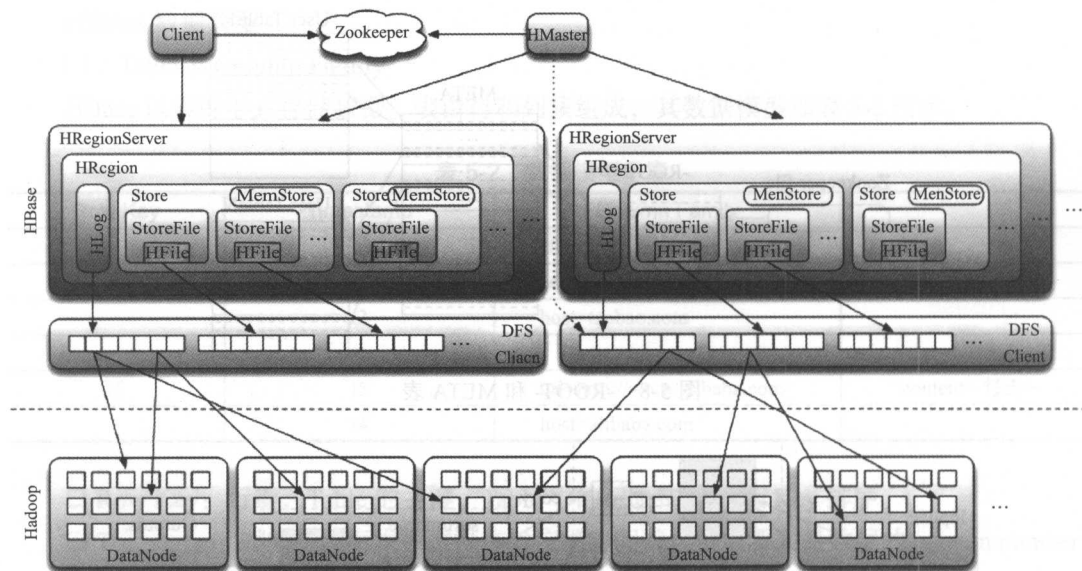


图 5-10 HBase 底层架构

（3）HMaster

HMaster 没有单点问题，HBase 中可以启动多个 HMaster，通过 Zookeeper 的 Master Election 机制保证总有一个 Master 运行，HMaster 在功能上主要负责 Table 和 Region 的管理工作：

- ❑ 管理用户对 Table 的增、删、改、查操作。
- ❑ 管理 HRegionServer 的负载均衡，调整 Region 分布。
- ❑ 在 Region Split 后，负责新 Region 的分配。
- ❑ 在 HRegionServer 停机后，负责失效 HRegionServer 上的 Regions 迁移。

（4）HRegionServer

HRegionServer 主要负责响应用户 I/O 请求，向 HDFS 文件系统中读写数据是 HBase 中最核心的模块。

如图 5-11 所示，HRegionServer 内部管理了一系列 HRegion 对象，每个 HRegion 对应了 Table 中的一个 Region，HRegion 由多个 HStore 组成。每个 HStore 对应 Table 中的一个 Column Family 的存储，可以看出每个 Column Family 其实就是一个集中的存储单元，因此最好将具备共同 IO 特性的 column 放在一个 Column Family 中，这样最高效。

（5）HStore

HStore 存储是 HBase 存储的核心，其中由两部分组成，一部分是 MemStore，另一部分是 StoreFiles。MemStore 是 Sorted Memory Buffer，用户写入的数据首先会放入 MemStore，当 MemStore 满了以后会 Flush 成一个 StoreFile（底层实现是 HFile），当 StoreFile 文件数量增长到一定阈值，会触发 Compact 合并操作，将多个 StoreFiles 合并成一个 StoreFile，合并过程中会进行版本合并和数据删除，因此可以看出 HBase 其实只有增加数据，所有的更新

和删除操作都是在后续的 Compact 过程中进行的,这使得用户的写操作只要进入内存中就可以立即返回,保证了 HBase I/O 的高性能。当 StoreFiles Compact 后,会逐步形成越来越大的 StoreFile,当单个 StoreFile 大小超过一定阈值后,会触发 Split 操作,同时把当前 Region Split 成 2 个 Region,父 Region 会下线,新 Split 出的 2 个孩子 Region 会被 HMaster 分配到相应的 HRegionServer 上,使得原先 1 个 Region 的压力得以分流到 2 个 Region 上。

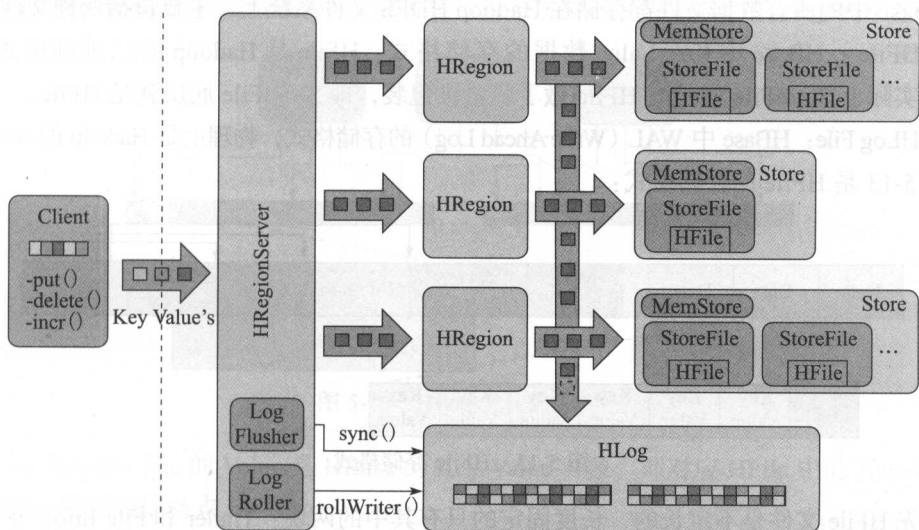


图 5-11 HRegionServer 响应

图 5-12 描述了 Compaction 和 Split 的过程:

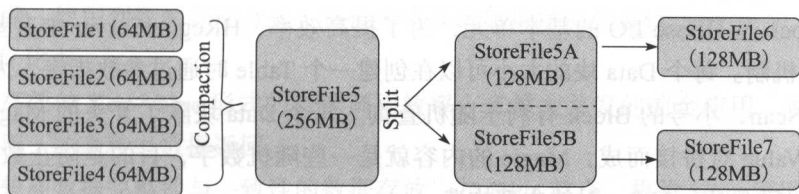


图 5-12 Compaction 和 Split 过程

在理解了上述 HStore 的基本原理后,还必须了解一下 HLog 的功能,因为上述 HStore 在系统正常工作的前提下是没有问题的,但是在分布式系统环境中,无法避免系统出错或者宕机,因此一旦 HRegionServer 意外退出,MemStore 中的内存数据将会丢失,这就需要引入 HLog 了。

每个 HRegionServer 中都有一个 HLog 对象,HLog 是一个实现 Write Ahead Log 的类,在每次用户操作写入 MemStore 的同时,也会写一份数据到 HLog 文件中(HLog 文件格式见下文),HLog 文件定期会滚动出新的,并删除旧的文件(已持久化到 StoreFile 中的数据)。

当 HRegionServer 意外终止后,HMaster 会通过 Zookeeper 感知到,HMaster 首先会处理

遗留的 HLog 文件，将其中不同 Region 的 Log 数据进行拆分，分别放到相应 Region 的目录下，然后再将失效的 Region 重新分配，领取到这些 Region 的 HRegionServer 在 Load Region 的过程中，会发现有历史 HLog 需要处理，因此会 Replay HLog 中的数据到 MemStore 中，然后 Flush 到 StoreFiles，完成数据恢复。

HBase 存储格式

HBase 中的所有数据文件都存储在 Hadoop HDFS 文件系统上，主要包括两种文件类型：

□ HFile：HBase 中 Key-Value 数据的存储格式，HFile 是 Hadoop 的二进制格式文件，实际上 StoreFile 就是对 HFile 做了轻量级包装，即 StoreFile 底层就是 HFile。

□ HLog File：HBase 中 WAL (Write Ahead Log) 的存储格式，物理上是 Hadoop 的 Sequence。

图 5-13 是 HFile 的存储格式：

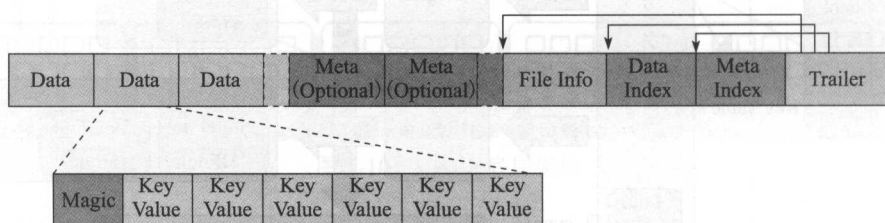


图 5-13 HFile 存储格式

首先 HFile 文件是不定长的，长度固定的只有其中的两块：Trailer 和 File Info。正如图中所示，Trailer 中有指针指向其他数据块的起始点。File Info 中记录了一些 Meta 信息，例如：AVG_KEY_LEN、AVG_VALUE_LEN、LAST_KEY、COMPARATOR、MAX_SEQ_ID_KEY 等。Data Index 和 Meta Index 块记录了每个 Data 块和 Meta 块的起始点。

Data Block 是 HBase I/O 的基本单元，为了提高效率，HRegionServer 中有基于 LRU 的 Block Cache 机制。每个 Data 块的大小可以在创建一个 Table 时通过参数指定，大号的 Block 有利于顺序 Scan，小号的 Block 有利于随机查询。每个 Data 块除了开头的 Magic 以外就是一个 Key-Value 对拼接而成，Magic 的内容就是一些随机数字，目的是防止数据损坏。下面会详细介绍每个 Key-Value 对的内部构造。

HFile 中的每个 Key-Value 对就是一个简单的 byte 数组。但是这个 byte 数组包含了很多项，并且有固定的结构。我们来看看里面的具体结构，如图 5-14 所示。

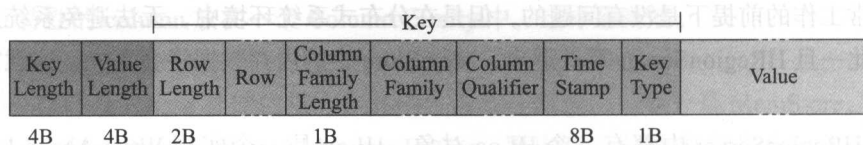


图 5-14 Key Value 存储结构

开始是两个固定长度的数值，分别表示 Key 的长度和 Value 的长度。紧接着是 Key，开始是固定长度的数值，表示 RowKey 的长度，紧接着是 RowKey，然后是固定长度的数值，表

示 Family 的长度, 然后是 Family, 接着是 Qualifier, 然后是两个固定长度的数值, 表示 Time Stamp 和 Key Type (Put/Delete)。Value 部分没有这么复杂的结构, 就是纯粹的二进制数据了。

图 5-15 中示意了 HLog 文件的结构, 其实 HLog 文件就是一个普通的 Hadoop Sequence File, Sequence File 的 Key 是 HLogKey 对象, HLogKey 中记录了写入数据的归属信息, 除了 Table 和 Region 名字外, 同时还包括 Sequence Number 和 Timestamp, Timestamp 是“写入时间”, Sequence Number 的起始值为 0, 或者是最近一次存入文件系统中的 Sequence Number。

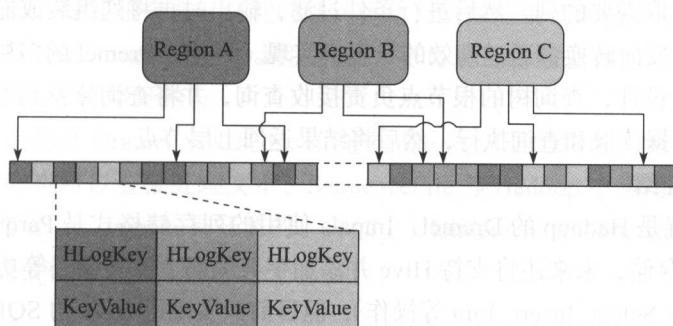


图 5-15 HLog 文件的结构

HLog Sequence File 的 Value 是 HBase 的 Key-Value 对象, 即对应 HFile 中的 Key-Value。

HBase 数据存储特点

数据存储使用 HBase 来承接, HBase 的功能特点:

- ❑ 高可靠: 可使用 Zookeeper 进行管理, 提供稳定性与一致性的保证。
- ❑ 高性能: 使用 Key-Value 的数据存储方式, 提供数据的快速存储与提取功能。
- ❑ 高伸缩: 使用列簇的存储方式, 提供非常高的数据横向扩展性。

性能的优化点:

- ❑ 提供高效的 Key-Value 形式的数据调用效率, 支撑大并发的前台应用。可支持 1W 并发调用下的毫秒级数据返回。
- ❑ 提供较高数据完整性与一致性的数据存放, 与 Hive 整合, 提供 Hive 调用 HBase 数据功能, 提升数据的准确率。
- ❑ 根据需求提供高效的 HBase 二级索引功能。提高复杂查询需求的查询效率。

5. Impala

Impala 是 Cloudera 公司主导开发的新型查询系统, 它提供 SQL 语义, 能查询存储在 Hadoop 的 HDFS 和 HBase 中的 PB 级大数据。已有的 Hive 系统虽然也提供了 SQL 语义, 但由于 Hive 底层执行使用的是 MapReduce 引擎, 仍然是一个批处理过程, 难以满足查询的交互性。相比之下, Impala 的最大特点也是最大卖点就是它的快速。那么 Impala 如何实现大数据的快速查询呢? 在回答这个问题前, 需要先介绍 Google 的 Dremel 系统, 因为 Impala 最开始是参照 Dremel 系统进行设计的。

Dremel 是 Google 的交互式数据分析系统，它构建于 Google 的 GFS (Google File System) 等系统之上，支撑了 Google 的数据分析服务 BigQuery 等诸多服务。Dremel 的技术亮点主要有两个：一是实现了嵌套型数据的列存储；二是使用了多层查询树，使得任务可以在数千个节点上并行执行和聚合结果。列存储在关系型数据库中并不陌生，它可以减少查询时处理的数据量，有效提升查询效率。Dremel 的列存储的不同之处在于它针对的并不是传统的关系数据，而是嵌套结构的数据。Dremel 可以将一条条的嵌套结构的记录转换成列存储形式，查询时根据查询条件读取需要的列，然后进行条件过滤，输出时再将列组装成嵌套结构的记录输出，记录的正向和反向转换都通过高效的状态机实现。另外，Dremel 的多层查询树则借鉴了分布式搜索引擎的设计，查询树的根节点负责接收查询，并将查询分发到下一层节点，底层节点负责具体的数据读取和查询执行，然后将结果返回上层节点。

Impala 架构介绍

Impala 其实就是 Hadoop 的 Dremel，Impala 使用的列存储格式是 Parquet。Parquet 实现了 Dremel 中的列存储，未来还将支持 Hive 并添加字典编码、游程编码等功能。使用了 Hive 的 SQL 接口（包括 Select、Insert、Join 等操作），但目前只实现了 Hive 的 SQL 语义的子集（例如尚未对 UDF 提供支持），表的元数据信息存储在 Hive 的 Metastore 中。

StateStore 是 Impala 的一个子服务，用来监控集群中各个节点的健康状况，提供节点注册、错误检测等功能。Impala 在每个节点运行了一个后台服务 Impalad，Impalad 用来响应外部请求，并完成实际的查询处理。

Impalad 主要包含 Query Planner、Query Coordinator 和 Query Exec Engine 三个模块。Query Planner 接收来自 SQL APP 和 ODBC 的查询，然后将查询转换为许多子查询，Query Coordinator 将这些子查询分发到各个节点上，由各个节点上的 Query Exec Engine 负责子查询的执行，最后返回子查询的结果，这些中间结果经过聚集之后最终返回给用户。系统架构如图 5-16 所示。

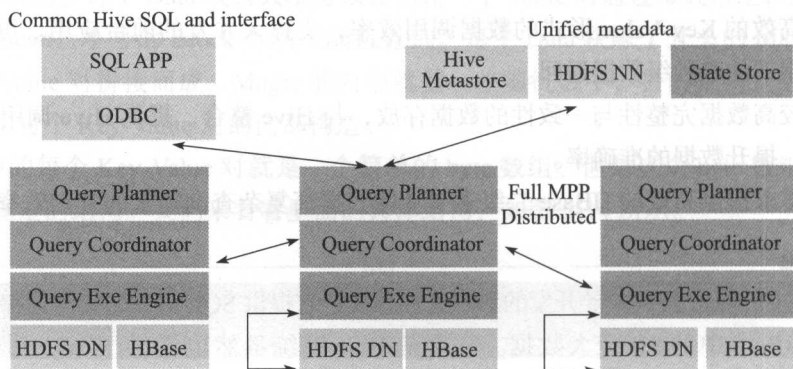


图 5-16 架构图

Impala 特点

□ Impala 不需要把中间结果写入磁盘，省掉了大量的 I/O 开销。

- 省掉了 MapReduce 作业启动的开销。MapReduce 启动 task 的速度很慢（默认每个心跳间隔是 3 秒钟），Impala 直接通过相应的服务进程来进行作业调度，速度快了很多。
- Impala 完全抛弃了 MapReduce 这个不太适合做 SQL 查询的范式，而是像 Dremel 一样借鉴了 MPP 并行数据库的思想另起炉灶，因此可做更多的查询优化，从而省掉不必要的 Shuffle、Sort 等开销。
- 通过使用 LLVM 来统一编译运行时代码，避免了为支持通用编译而带来的不必要开销。
- 用 C++ 实现，做了很多有针对性的硬件优化，例如使用 SSE 指令。
- 使用了支持 Data locality 的 I/O 调度机制，尽可能地将数据和计算分配在同一台机器上进行，减少了网络开销。

虽然 Impala 是参照 Dremel 来实现的，但它也有一些自己的特色，例如 Impala 不仅支持 Parquet 格式，同时也可以直接处理文本、SequenceFile 等 Hadoop 中常用的文件格式。另外一个更关键的地方在于，Impala 是开源的，再加上 Cloudera 在 Hadoop 领域的领导地位，其生态圈有很大可能会在将来快速成长。

Impala 设计

Impala 的设计类似于商用并行关系数据库，该分布式查询引擎由 Query Planner、Query Coordinator 和 Query Exec Engine 三部分组成。运行过程中主要有两大重要进程：State Store 与 Impalad。下面通过对这两大进程的分析，说明 Impala 的运行流程。

□ State Store：用于协调各个运行 Impalad 的实例之间的信息关系，Impala 正是通过这些信息去定位查询请求所要的数据。换句话说，State Store 的作用主要为跟踪各个 Impalad 实例的位置和状态，让各个 Impalad 实例以集群的方式运行起来。

与 HDFS 的 NameNode 不一样，虽然 StateStore 一般只安装一份，但一旦 StateStore 挂掉了，各个 Impalad 实例却仍然会保持集群的方式处理查询请求，只是无法将各自的状态更新到 StateStore 中，如果此时新加入一个 Impalad 实例，则新加入的 Impalad 实例不为现有集群中的其他 Impalad 实例所识别。然而，StateStore 一旦重启，则所有 StateStore 所服务的各个 Impalad 实例（包括 State Store 挂掉期间新加入的 Impalad 实例）的信息（由 Impalad 实例发给 StateStore）都会进行重建。

□ Impalad：对应进程为 Impalad（核心进程，数据的计算就靠这个进程来执行），该进程应运行在 DataNode 机器上，每个 DataNode 机器运行一个 Impalad，每个 Impalad 实例会接收、规划并调节来自 ODBC 或 Impala Shell 等客户端的查询。每个 Impalad 实例会充当一个 Worker，处理由其他 Impalad 实例分发出来的查询片段（Query Fragments）。客户端可以随便连接到任意一个 Impalad 实例，被连接的 Impalad 实例将充当本次查询的协调者（Ordinator），将查询分发给集群内的其他 Impalad 实例进行并行计算。当所有计算完毕时，其他各个 Impalad 实例将会把各自的计算结果发送给充当 Ordinator 的 Impalad 实例，由这个 Ordinator 实例把结果返回给客户端。每个 Impalad 进程可以处理多个并发请求。

Impala 与 Hive 对比

虽然 Impala 使用了 Hive 的元数据，但是在架构和使用上还是存在很大的区别，如表 5-3 所示是 Impala 与 Hive 的主要区别。

表 5-3 Impala 与 SparkSQL 和 Apache Drill 等同类型组件的比较

	Impala	Hive
定义	在 Hadoop 集群上运行的本地 SQL 查询引擎，提供原始 HDFS 数据和 HBase 数据库的简单查询访问	建立在 Hadoop 上的数据仓库框架
出品人	Cloudera	Apache
发布时间	2012 年 10 月	
Impala 1.0 beta 版		
与 MapReduce 的关系	无需通过 MapReduce 进行计算	MapReduce+Hive 计算方式
与 HDFS、HBase 的关系	可以直接从 HDFS 或者 HBase 中用 Select、Join 和统计函数查询数据	与 HDFS、HBase 可以联合部署
查询语言	HQL	HiveQL
UDF 支持	1.0 不支持	支持
中间结果的处理	不存储磁盘	存储磁盘
最佳负载类型	批量提取、转化、加载 (ETL) 类型的 Job	实时查询
两者的关系	Impala 的运行依赖 Hive 的元数据	

开源组织 Apache 也发起了名为 Drill 的项目来实现 Hadoop 上的 Dremel，目前该项目正在开发中，相关的文档和代码还不多，暂时还未对 Impala 构成足够的威胁。从 Quora 上的问答来看，Cloudera 有 7 ~ 8 名工程师全职在 Impala 项目上，而相比之下 Drill 目前的动作稍显迟钝。

具体来说，截至 2012 年 10 月底，Drill 的代码库里实现了 query parser、plan parser 及能对 JSON 格式的数据进行扫描的 plan evaluator；而 Impala 同期已经有了一个比较完备的分布式 query execution 引擎，并对 HDFS 和 HBase 上的数据读入、错误检测、INSERT 的数据修改、LLVM 动态翻译等都提供了支持。当然，Drill 作为 Apache 的项目，从一开始就避免了某个厂商的一家独大，而且对所有 Hadoop 流行的发行版都会做相应的支持，不像 Impala 只支持 Cloudera 自己的发行版 CDH。从长远来看，谁会占据上风还真不一定。

除此之外，加州伯克利大学 AMPLab 也开发了名为 Shark 的大数据分析系统。从长远目标来看，SparkSQL 想成为一个既支持大数据 SQL 查询，又能支持高级数据分析任务的一体化数据处理系统。从技术实现的角度上来看，SparkSQL 基于 Scala 语言的算子推导实现了良好的容错机制，因此对失败了的长任务和短任务都能从上一个“快照点”进行快速恢复。

相比之下，Impala 由于缺失足够强大的容错机制，其上运行的任务一旦失败就必须“从头来过”，这样的设计必然会在性能上有所缺失，而且 Shark 是把内存当做第一类的存储介质来做的系统设计，所以在处理速度上也会有一些优势。

实际上，AMPLab 最近对 Hive、Impala、Shark 及 Amazon 采用的商业 MPP 数据库 Redshift

进行了一次对比试验，在 Scan Query、Aggregation Query 和 Join Query 三种类型的任务中对它们进行了比较。图 5-17 就是 AMPLab 报告中 Aggregation Query 的性能对比。从图中我们可以看到，商业版本的 Redshift 的性能是最好的，Impala 和 SparkSQL 则各有胜负，且两者都比 Hive 的性能高出了一大截。其实对大数据分析的项目来说，技术往往不是最关键的，例如 Hadoop 中的 MapReduce 和 HDFS 都是源于 Google，原创性较少。

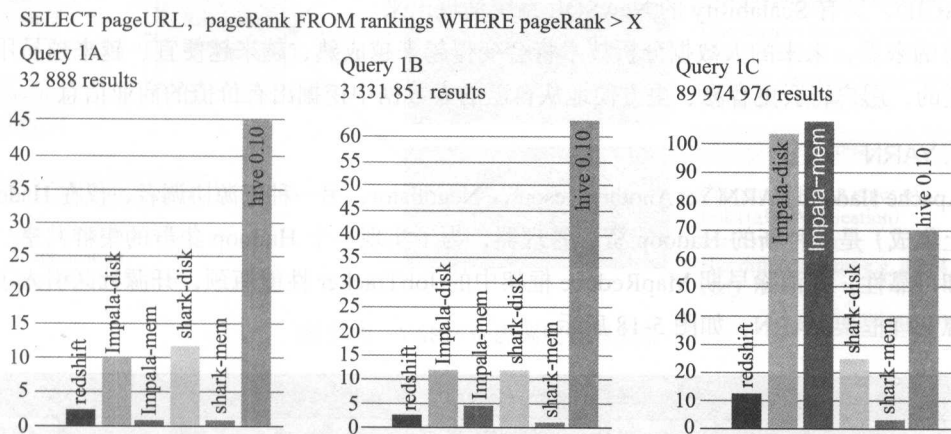


图 5-17 Redshift、Impala、SparkSQL 与 Hive 的 Aggregation Query 性能对比

事实上，开源项目的生态圈、社区、发展速度等，往往在很大程度上会影响 Impala 和 Shark 等开源大数据分析系统的发展。就像 Cloudera 一开始就决定会把 Impala 开源，以期望利用开源社区的力量来推广这个产品；SparkSQL 也是一开始就开源了出来，更不用说 Apache 的 Drill 更是如此。说到底还是谁的生态系统更强的问题。技术上一时的领先并不足以保证项目的最终成功。虽然最后哪一款产品会成为事实上的标准还很难说，但我们唯一可以确定并坚信的一点是，大数据分析将随着新技术的不断推陈出新而不断普及开来，这对用户永远都是一件幸事。

未来展望

可以预见，在不久的将来，Impala 很可能像之前的 Hadoop 和 Hive 一样在大数据处理领域大展拳脚。Cloudera 自己也说期待未来 Impala 能完全取代 Hive。当然，用户从 Hive 上迁移到 Impala 上是需要时间的，而且 Impala 也只是刚刚发布 1.0 版，虽然号称已经可以稳定地在生产环境上运行，但相信仍然有很多可改进的空间。需要说明的是，Impala 并不是用来取代已有的 MapReduce 系统，而是作为 MapReduce 的一个强力补充。

Impala 适合用来处理输出数据适中或比较小的查询，而对于大数据量的批处理任 MapReduce 依然是更好的选择。另外一个消息是，Cloudera 里负责 Impala 的架构师 Marcel Komacker 就曾在 Google 负责过 F1 系统的查询引擎开发，可见 Google 确实为大数据的流行出钱出力。

其实除了 Impala、Shark 和 Drill 这样的开源方案外，像 Oracle、EMC 等传统厂商也没在坐以待毙等着自己的市场被开源软件侵吞。比如，EMC 就推出了 HAWQ 系统，并号称其性能比 Impala 快上十几倍，而前面提到的 Amazon 的 Redshift 也提供了比 Impala 更好的性能。

虽然开源软件因为其强大的成本优势而拥有极其强大的力量，但传统数据库厂商仍会尝试推出在性能、稳定性、维护服务等指标上更加强大的产品与之进行差异化竞争，并同时通过参与开源社区、借力开源软件来丰富自己的产品线，提升自己的竞争力，并通过更多的高附加值服务来满足某些消费者需求。毕竟，这些厂商往往已在并行数据库等传统领域积累了大量的技术和经验，底蕴非常深厚。甚至现在还有像 NuoDB（一个创业公司）这样号称既支持 ACID，又有 Scalability 的 NewSQL 系统涌现出来。

总的来看，未来的大数据分析技术将会变得越来越成熟、越来越便宜、越来越易用。与此相应的，用户将会更容易、更方便地从自己的大数据中挖掘出有价值的商业信息。

6. YARN

Apache Hadoop YARN(Yet Another Resource Negotiator，另一种资源协调者，仅在 Hadoop2.0 版以上集成)是一种新的 Hadoop 资源管理器，为了实现一个 Hadoop 集群的集群共享、可伸缩性和可靠性，并消除早期 MapReduce 框架中的 JobTracker 性能瓶颈，开源社区引入了统一的资源管理框架 YARN，如图 5-18 所示。

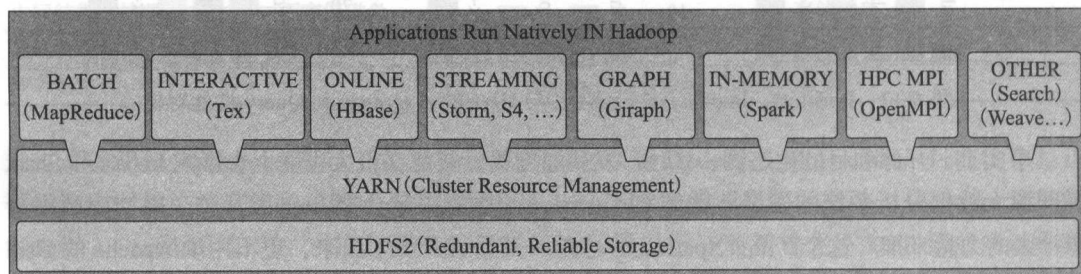


图 5-18 YARN 架构图

YARN 是基于 HDFS 上层的一个资源调度管理系统，只提供资源申请调度相关的接口，任何 Client 都可以通过资源接口申请资源运行在 YARN 之上，例如 MapReduce、HBase、Spark 等。

下面是 YARN 几个组件之间的关系，如图 5-19 所示。

在 YARN 架构中，一个全局 ResourceManager 主要以后台进程的形式运行，它通常在专用机器上运行，在各种竞争的应用程序之间仲裁可用的集群资源。ResourceManager 会追踪集群中有多少可用的活动节点和资源，协调用户提交的哪些应用程序应该在何时获取这些资源。ResourceManager 是唯一拥有此信息的进程，所以它可通过某种共享的、安全的、多租户的方式制定分配（或者调度）决策（例如依据应用程序优先级、队列容量、ACLs、数据位置等）。

在用户提交一个应用程序时，一个称为 ApplicationMaster 的轻量型进程实例会启动来协调应用程序内的所有任务的执行。这包括监视任务、重新启动失败的任务、推测性地运行缓慢的任务，以及计算应用程序计数器值的总和。这些职责以前分配给所有作业的单个 JobTracker。ApplicationMaster 和属于它的应用程序的任务，在受 NodeManager 控制的资源容器中运行。

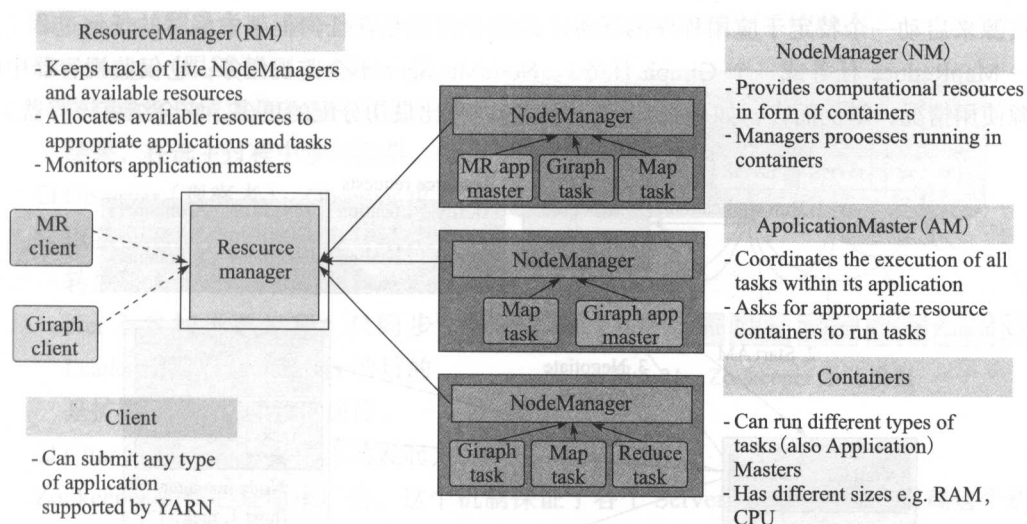


图 5-19 YARN 资源调度管理关系

NodeManager 是 TaskTracker 的一种更加普通和高效的版本。没有固定数量的 map 和 reduce slots, NodeManager 拥有许多动态创建的资源容器。容器的大小取决于它所包含的资源量, 例如内存、CPU、磁盘和网络 IO。目前, 仅支持内存和 CPU (YARN-3)。未来可使用 cgroups 来控制磁盘和网络 IO。一个节点上的容器数量, 由配置参数与专用于从属后台进程和操作系统的资源以外的节点资源总量 (例如总 CPU 数和总内存) 共同决定。

假设用户采用键入 `hadoop jar` 命令的方式, 将应用程序提交到 ResourceManager。ResourceManager 维护在集群上运行的应用程序列表, 以及每个活动的 NodeManager 上的可用资源列表。ResourceManager 需要确定哪个应用程序接下来应该获得一部分集群资源, 如图 5-20 所示。该决策受到许多限制, 例如队列容量、ACL 和公平性。ResourceManager 使用一个可插拔的 Scheduler。Scheduler 仅执行调度; 它管理谁在何时获取集群资源 (以容器的形式), 但不会对应用程序内的任务执行任何监视, 所以它不会尝试重新启动失败的任务。

在 ResourceManager 接受一个新应用程序提交时, Scheduler 制定的第一个决策是选择将用来运行 ApplicationMaster 的容器。在 ApplicationMaster 启动后, 它将负责此应用程序的整个生命周期。首先也是最重要的是, 它将资源请求发送到 ResourceManager, 请求运行应用程序的任务所需的容器。资源请求是对一些容器的请求, 用以满足一些资源需求, 例如:

- 一定量的资源, 目前使用 MB 内存和 CPU 份额来表示。
- 一个首选的位置, 由主机名、机架名称指定, 或者使用 * 来表示没有偏好。
- 此应用程序中的一个优先级, 而不是跨多个应用程序。

如果可能的话, ResourceManager 会分配一个满足 ApplicationMaster 在资源请求中所请求的需求的容器 (表达为容器 ID 和主机名)。该容器允许应用程序使用特定主机上给定的资源量。分配一个容器后, ApplicationMaster 会要求 NodeManager (管理分配容器的主机) 使用这

些资源来启动一个特定于应用程序的任务。此任务可以在任何框架中编写的任何进程（例如一 MapReduce 任务或一个 Giraph 任务）。NodeManager 不会监视任务；它仅监视容器中的资源使用情况，举例而言，如果一个容器消耗的内存比最初分配的更多，它会结束该容器。

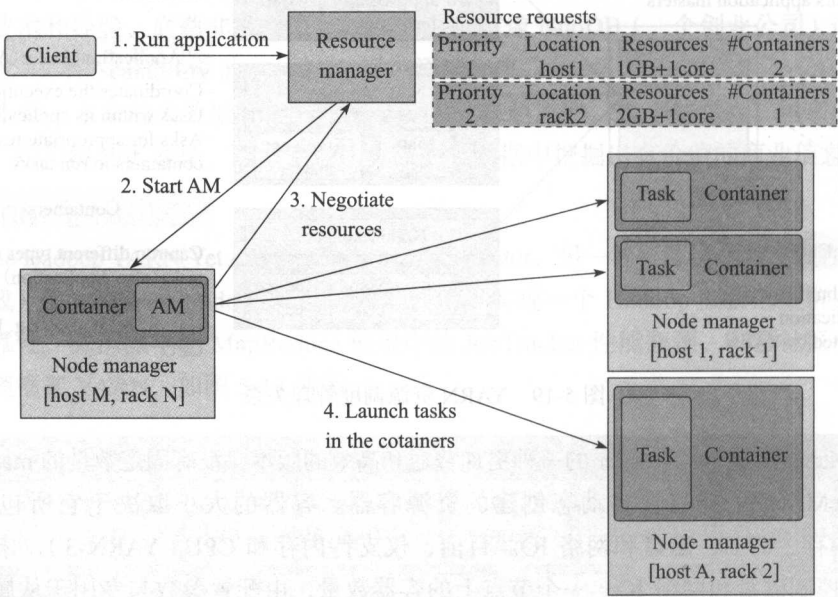


图 5-20 YARN 任务提交流程

ApplicationMaster 会竭尽全力协调容器，启动所有需要的任务来完成它的应用程序。它还监视应用程序及其任务的进度，在新请求的容器中重新启动失败的任务，以及向提交应用程序的客户端报告进度。应用程序完成后，ApplicationMaster 会关闭自己并释放自己的容器。

尽管 ResourceManager 不会对应用程序内的任务执行任何监视，但它会检查 ApplicationMaster 的健康状况。如果 ApplicationMaster 失败，ResourceManager 可在一个新容器中重新启动它。您可以认为 ResourceManager 负责管理 ApplicationMaster，而 ApplicationMasters 负责管理任务。

7. Zookeeper

Zookeeper 是一个分布式的、开放源码的分布式应用程序协调服务，是 Google 的 Chubby 一个开源的实现，是保障 Hadoop 和 HBase 正常运行的重要组件。它为分布式应用提供一致性服务，提供的功能包括：配置维护、名字服务、分布式同步、组服务等。其目标就是封装好复杂易出错的关键服务，将简单易用的接口和性能高效、功能稳定的系统提供给用户。Zookeeper 包含一个简单的原语集并提供 Java 和 C 的接口，在源码版本中，还提供了分布式独享锁、选举、队列的接口。Zookeeper 中分为几个角色，分别为领导者（Leader）、学习者（Learner 又分为跟随者（Follower）和观察者（Observer）、客户端。系统模型如图 5-21 所示。

❑ Leader (领导者): 领导者负责进行投票的发起和决议, 更新系统状态。

❑ Follower (跟随者): Follower 用户接收客户端请求并向客户端返回结果, 在选主过程中参与投票。

❑ Observer (观察者): Observer 可以接收客户端连接, 将写请求转发给 Leader 节点, 但是 Observer 不参加投票过程, 只同步 Leader 的状态, Observer 的目的是扩展系统, 提供读取速度。

❑ Client (客户端): 是请求的发起方。

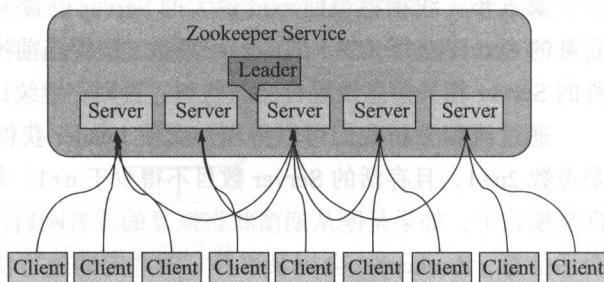


图 5-21 Zookeeper 系统模型

Zookeeper 的核心是原子广播, 这个机制保证了各个 Server 之间的同步。实现这个机制的协议叫做 Zab 协议。Zab 协议有两种模式, 它们分别是恢复模式 (选主) 和广播模式 (同步)。当服务启动或者在领导者崩溃后, Zab 就进入了恢复模式, 当领导者被选举出来, 且大多数 Server 完成了和 Leader 的状态同步以后, 恢复模式就结束了。状态同步保证了 Leader 和 Server 具有相同的系统状态。

为了保证事务的顺序一致性, Zookeeper 采用了递增的事务 id 号 (zxid) 来标识事务。所有的提议 (Proposal) 都在被提出时加上了 zxid。实现中 zxid 是一个 64 位的数字, 高 32 位是 epoch 用来标识 Leader 关系是否改变, 每次一个 Leader 被选出来, 它都会有一个新的 epoch, 标识当前属于那个 Leader 的统治时期。低 32 位用于递增计数。每个 Server 在工作过程中有三种状态:

❑ LOOKING: 当前 Server 不知道 Leader 是谁, 正在搜寻。

❑ LEADING: 当前 Server 即为选举出来的 Leader。

❑ FOLLOWING: Leader 已经选举出来, 当前 Server 与之同步。

当 Leader 崩溃或者 Leader 失去大多数的 Follower, 此时 Zookeeper 进入恢复模式, 恢复模式需要重新选举出一个新的 Leader, 让所有的 Server 都恢复到一个正确的状态。Zookeeper 的选举算法有两种: 一种是基于 basic paxos 算法实现的, 另外一种是基于 fast paxos 算法实现的。系统默认的选举算法为 fast paxos。先介绍 basic paxos 流程:

第一步: 选举线程由当前 Server 发起选举的线程担任, 其主要功能是对投票结果进行统计, 并选出推荐的 Server;

第二步: 选举线程首先向所有 Server 发起一次询问 (包括自己);

第三步: 选举线程收到回复后, 验证是否是自己发起的询问 (验证 zxid 是否一致), 然后获取对方的 id (myid), 并存储到当前询问对象列表中, 最后获取对方提议的 Leader 相关信息 (id, zxid), 并将这些信息存储到当次选举的投票记录表中;

第四步: 收到所有 Server 回复以后, 就计算出 zxid 最大的那个 Server, 并将这个 Server

相关信息设置成下一次要投票的 Server；

第五步：线程将当前 zxid 最大的 Server 设置为当前 Server 要推荐的 Leader，如果此时获胜的 Server 获得 $n/2+1$ 的 Server 票数，设置当前推荐的 Leader 为获胜的 Server，将根据获胜的 Server 相关信息设置自己的状态，否则，继续这个过程，直到 Leader 被选举出来。

通过流程分析我们可以得出：要使 Leader 获得多数 Server 的支持，则 Server 总数必须是奇数 $2n+1$ ，且存活的 Server 数目不得少于 $n+1$ 。每个 Server 启动后都会重复以上流程。在恢复模式下，如果是刚从崩溃状态恢复的或者刚启动的 Server 还会从磁盘快照中恢复数据和会话信息，Zookeeper 会记录事务日志并定期进行快照，方便在恢复时进行状态恢复。选举的具体流程如图 5-22 所示。

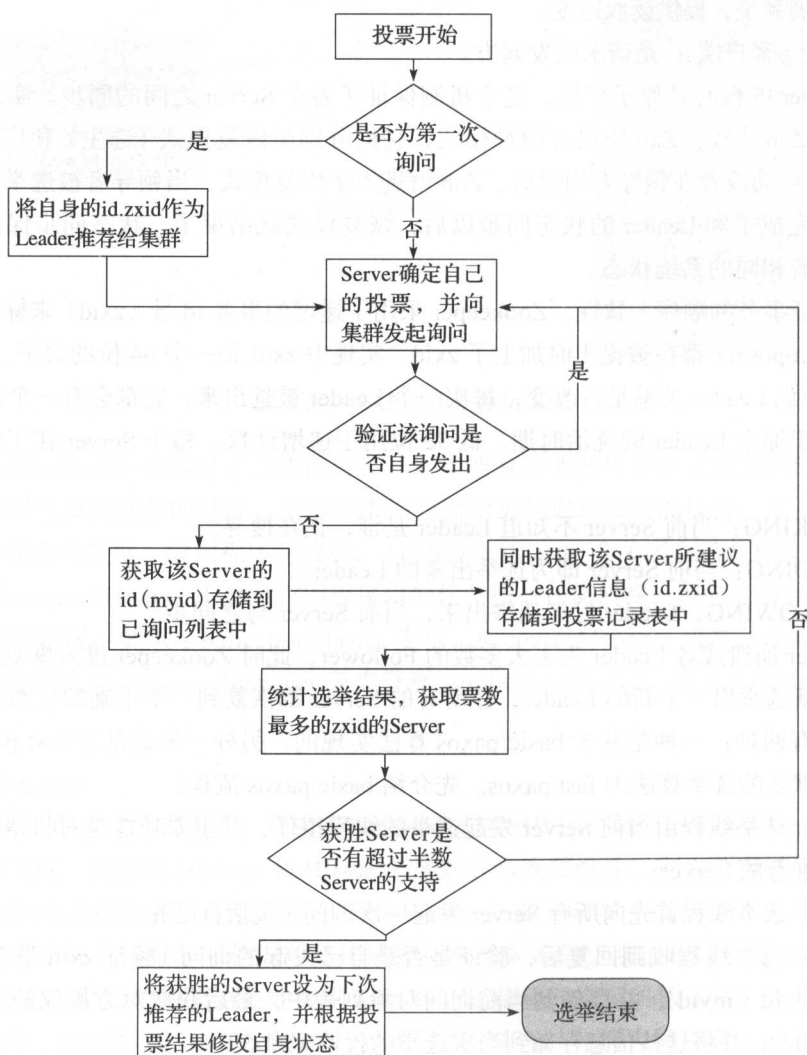


图 5-22 Zookeeper 选举 Leader 流程

fast paxos 流程是在选举过程中, 某 Server 首先向所有 Server 提议自己要成为 Leader, 当其他 Server 收到提议以后, 解决 epoch 和 zxid 的冲突, 并接受对方的提议, 然后向对方发送接受提议完成的消息, 重复这个流程, 最后一定能选举出 Leader。其流程如图 5-23 所示。

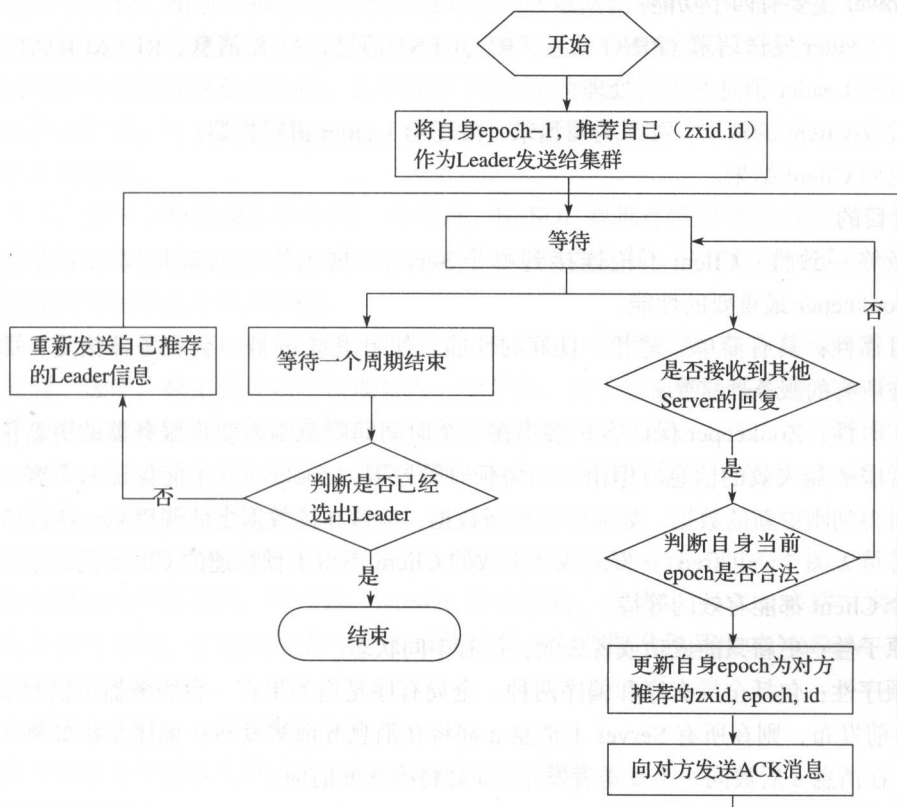


图 5-23 fast paxos 流程

选完 Leader 以后, Zookeeper 就进入状态同步过程。

❑ Leader 等待 Server 连接;

❑ Follower 连接 Leader, 将最大的 zxid 发送给 Leader;

❑ Leader 根据 Follower 的 zxid 确定同步点;

❑ 完成同步后通知 Follower 已经成为 uptodate 状态;

❑ Follower 收到 uptodate 消息后, 又可以重新接受 Client 的请求进行服务了。

Leader 主要有三个功能:

❑ 恢复数据;

❑ 维持与 Learner 的心跳, 接收 Learner 请求并判断 Learner 的请求消息类型;

❑ Learner 的消息类型主要有 PING 消息、REQUEST 消息、ACK 消息、REVALIDATE 消息, 根据不同的消息类型, 进行不同的处理。

其中，PING 消息是指 Learner 的心跳信息；REQUEST 消息是 Follower 发送的提议信息，包括写请求及同步请求；ACK 消息是 Follower 对提议的回复，超过半数的 Follower 通过，则 commit 该提议；REVALIDATE 消息是用来延长 SESSION 有效时间。

Follower 主要有四个功能：

- 向 Leader 发送请求（PING 消息、REQUEST 消息、ACK 消息、REVALIDATE 消息）；
- 接收 Leader 消息并进行处理；
- 接收 Client 的请求，如果为写请求，发送给 Leader 进行投票；
- 返回 Client 结果。

设计目的

- 最终一致性：Client 不论连接到哪个 Server，展示给它的都是同一个视图，这是 Zookeeper 最重要的性能。
- 可靠性：具有简单、健壮、良好的性能，如果消息 m 被一台服务器接受，那么它将被所有的服务器接受。
- 实时性：Zookeeper 保证客户端将在一个时间间隔范围内获得服务器的更新信息，或者服务器失效的信息。但由于网络延时等原因，Zookeeper 不能保证两个客户端能同时得到刚更新的数据，如果需要最新数据，应该在读数据之前调用 sync() 接口。
- 等待无关（wait-free）：慢的或者失效的 Client 不得干预快速的 Client 的请求，使得每个 Client 都能有效的等待。
- 原子性：更新只能成功或者失败，没有中间状态。
- 顺序性：包括全局有序和偏序两种。全局有序是指如果在一台服务器上消息 a 在消息 b 前发布，则在所有 Server 上消息 a 都将在消息 b 前被发布；偏序是指如果一个消息 b 在消息 a 后被同一个发送者发布，a 必将排在 b 前面。

5.1.2 NoSQL

NoSQL（NoSQL = Not Only SQL），意即“不仅仅是 SQL”，泛指非关系型的数据库。它是一项全新的数据库革命性运动，早期就有人提出，发展至 2009 年趋势越发高涨。随着互联网 Web2.0 网站的兴起，传统的关系数据库在应付 Web2.0 网站，特别是超大规模和高并发的 SNS 类型的 Web2.0 纯动态网站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。NoSQL 数据库的产生就是为了解决大规模数据集合多重数据种类带来的挑战，尤其是大数据应用难题。

尽管早期的堆栈代码只能算是一种实验，然而现在的系统已经更加成熟、稳定。不过现在也面临着一个严酷的事实：技术越来越成熟——以至于原来很好的 NoSQL 数据存储不得不进行重写，也有少数人认为这就是所谓的 2.0 版本。这里列出一些比较知名的工具，可以为大数据建立快速、可扩展的存储库。NoSQL 的拥护者提倡运用非关系型的数据存储，相对于铺天盖地的关系型数据库运用，这一概念无疑是一种全新思维的注入。

对于 NoSQL 并没有一个明确的范围和定义，但是它们都普遍存在下面一些共同特征：

- ❑ 不需要预定义模式：不需要事先定义数据模式，预定义表结构。数据中的每条记录都可能有不同的属性和格式。当插入数据时，并不需要预先定义它们的模式。
- ❑ 无共享架构：相对于将所有数据存储的存储区域网络中的全共享架构，NoSQL 往往将数据划分后存储在各个本地服务器上。因为从本地磁盘读取数据的性能往往好于通过网络传输读取数据的性能，从而提高了系统的性能。
- ❑ 弹性可扩展：可以在系统运行时，动态增加或者删除结点。不需要停机维护，数据可以自动迁移。
- ❑ 分区：相对于将数据存放于同一个节点，NoSQL 数据库需要将数据进行分区，将记录分散在多个节点上，并且通常分区的同时还要做复制。这样既提高了并行性能，又能保证没有单点失效的问题。
- ❑ 异步复制：和 RAID 存储系统不同的是，NoSQL 中的复制，往往是基于日志的异步复制。这样，数据就可以尽快地写入一个节点，而不会被网络传输引起迟延。缺点是并不总是能保证一致性，这样的方式在出现故障时，可能会丢失少量的数据。
- ❑ BASE：相对于事务严格的 ACID 特性，NoSQL 数据库保证的是 BASE 特性。BASE 是最终一致性和软事务。

NoSQL 数据库并没有一个统一的架构，两种 NoSQL 数据库之间的不同，甚至远远超过两种关系型数据库的不同。可以说，NoSQL 各有所长，成功的 NoSQL 必然特别适用于某些场合或者某些应用，在这些场合中会远远胜过关系型数据库和其他的 NoSQL。所以选择 NoSQL 数据库需要考虑：

- ❑ 数据的安全性 - 权限控制如何解决；
- ❑ 由于对事务性需求支持力度不够，事务性如何保障；
- ❑ 数据的重要性；
- ❑ 是否有 DBA 运维；
- ❑ 未来的业务需求变化。

1. MongoDB

MongoDB 是一个基于分布式文件存储的数据库，由 C++ 语言编写，旨在为 Web 应用提供可扩展的高性能数据存储解决方案。MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中最像关系数据库的。它支持的数据结构非常松散，类似 json 的 bson 格式，因此可以存储比较复杂的数据类型。Mongo 最大的特点是它支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

MongoDB 的特点

- ❑ 面向集合的存储：适合存储对象及 JSON 形式的数据。
- ❑ 动态查询：Mongo 支持丰富的查询表达式。查询指令使用 JSON 形式的标记，可轻易

查询文档中内嵌的对象及数组。

- ❑ 完整的索引支持：包括文档内嵌对象及数组。Mongo 的查询优化器会分析查询表达式，并生成一个高效的查询计划。
- ❑ 查询监视：Mongo 包含一个监视工具用于分析数据库操作的性能。
- ❑ 复制及自动故障转移：Mongo 数据库支持服务器之间的数据复制，支持主 - 从模式及服务器之间的相互复制。复制的主要目标是提供冗余及自动故障转移。
- ❑ 高效的传统存储方式：支持二进制数据及大型对象（例如照片或图片）。
- ❑ 自动分片以支持云级别的伸缩性：自动分片功能支持水平的数据库集群，可动态添加额外的机器。

MongoDB 适用场景

MongoDB 的适用场景如图 5-24 所示。

适用场景	网站数据	Mongo 非常适合实时地插入、更新与查询，并具备网站实时数据存储所需的复制及高度伸缩性
	缓存	由于性能很高，Mongo 也适合作为信息基础设施的缓存层。在系统重启之后，由 Mongo 搭建的持久化缓存层可以避免下层的数据源过载
	大尺寸低价值数据	使用传统的关系型数据库存储一些数据时存储成本可能会比较昂贵，在此之前，很多时候程序员往往会选择传统的文件进行存储
	高伸缩性	Mongo 非常适合由数十或数百台服务器组成的数据库。Mongo 的路线图中已经包含对 MapReduce 引擎的内置支持
	用于对象及 JSON 数据的存储	Mongo 的 BSON 数据格式非常适合文档化格式的存储及查询

图 5-24 MongoDB 适用场景

MongoDB 缺点

MongoDB 的缺点如图 5-25 所示。

缺点	数据量	在 32 位系统上，不支持大于 2.5G 的数据
	文档大小	单个文档大小限制为 16M（为了避免单个文档过大，完成读取时对内存和带宽占用过高，而影响查询性能和稳定性）
	权限控制	用户权限方面比较弱，将机器部署在安全的内网环境中，尽量不要用权限
	高度事务性的系统	比如，银行或会计系统。传统的关系型数据库目前还是更适用于需要大量原子性复杂事务的应用程序
	传统的商业智能应用	针对特定问题的 BI 数据库会产生高度优化的查询方式。对于此类应用，关系型数据仓库可能是更合适的选择

图 5-25 MongoDB 缺点

MongoDB 应用原理

MongoDB 集群包括一定数量的 mongod（分片存储数据）、mongos（路由处理）、config server（配置节点）、client（客户端）、arbiter（仲裁节点：为了选举某几个分片存储数据节点中哪个

为主节点)。如图 5-26 所示展示了 MongoDB 集群组成。

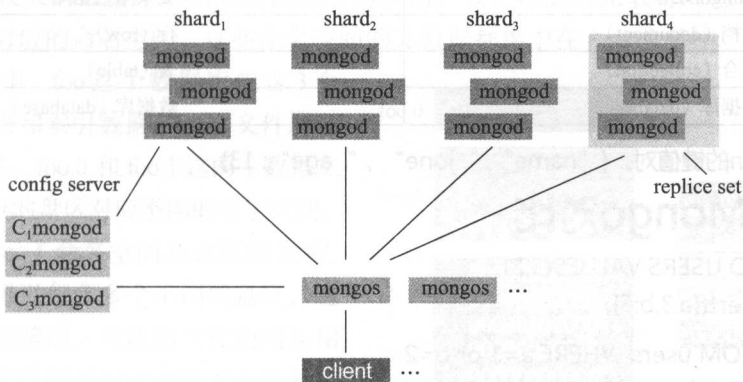


图 5-26 MongoDB 集群组成

- shard: 一个 shard 为一组 mongod，通常一组为两台，主从或互为主从，这一组 mongod 中的数据是相同的，具体可见《mongodb 分布式之数据复制》。数据分割按有序分割方式，每个分片上的数据为某一范围的数据块，故可支持指定分片的范围查询，这同 Google 的 BigTable 类似。数据块有指定的最大容量，一旦某个数据块的容量增长到最大容量时，这个数据块会切分成为两块；当分片的数据过多时，数据块将被迁移到系统的其他分片中。另外，新的分片加入时，数据块也会迁移。
- mongos: 可以有多个，相当于一个控制中心，负责路由和协调操作，使得集群像一个整体的系统。mongos 可以运行在任何一台服务器上，有些选择放在 shards 服务器上，也有放在 client 服务器上的。mongos 启动时需要从 config servers 上获取基本信息，然后接受 client 端的请求，路由到 shards 服务器上，然后整理返回的结果发回给 client 服务器。
- config server: 存储集群的信息，包括分片和块数据信息。主要存储块数据信息，每个 config server 上都有一份所有块数据信息的拷贝，以保证每台 config server 上数据的一致性。
- shard key: 为了分割数据集，需要制定分片 key 的格式，类似于用于索引的 key 格式，通常由一个或多个字段组成以分发数据，例如：

```
{ name : 1 }
{ _id : 1 }
{ lastname : 1, firstname : 1 }
{ tag : 1, timestamp : -1 }
```

MongoDB 的分片为有序存储（1 为升序，-1 为降序），shard key 相邻的数据通常会存在同一台服务器（数据块）上。

MongoDB 与关系型数据库逻辑结构对比如图 5-27 所示。

逻辑结构对比	
MongoDB	关系型数据库
文档 (document)	行 (row)
集合 (collection)	表 (table)
数据库 (database)	数据库 (database)

文档 类似于json的键值对。{ "name" : " jone" , " age" : 13}

SQL to Mongo对比

- INSERT INTO USERS VALUES(3,5)
db.users.insert({a:3,b:5})
- SELECT * FROM users WHERE a=1 or b=2
db.users.find({ \$or: [{ a : 1 } , { b : 2 }] })
- UPDATE users SET a=1 WHERE b='q' '
db.users.update({b:'q'}, {\$set:{a:1}}, false, true)

图 5-27 MongoDB 与关系型数据库对比

MongoDB 数据存储原理

MongoDB 的默认数据目录是 /data/db，它负责存储所有的 MongoDB 的数据文件。在 MongoDB 内部，每个数据库都包含一个 .ns 文件和一些数据文件，而且这些数据文件会随着数据量的增加而变得越来越多。所以如果系统中有一个叫做 foo 的数据库，那么构成 foo 这个数据库的文件就会由 foo.ns、foo.0、foo.1、foo.2 等组成，具体如下：

```
[root@localhostdb]# ll /data/db/
总计 196844
-rw----- 1 root root 16777216 04-15 16:33 admin.0
-rw----- 1 root root 33554432 04-15 16:33 admin.1
-rw----- 1 root root 16777216 04-15 16:33 admin.ns
-rw----- 1 root root 16777216 04-21 17:30 foo.0
-rw----- 1 root root 33554432 04-21 17:30 foo.1
-rw----- 1 root root 67108864 04-21 17:30 foo.2
-rw----- 1 root root 16777216 04-21 17:30 foo.ns -rwxr-xr-x 1 root root 6 04-
21 17:16 mongod.lock
-rw----- 1 root root 16777216 04-15 16:30 test.0
-rw----- 1 root root 33554432 04-15 16:30 test.1
-rw----- 1 root root 16777216 04-15 16:30 test.ns
drwxr-xr-x 2 root root 4096 04-21 17:30 _tmp
[root@localhostdb]#
```

MongoDB 内部有预分配表空间的机制，每个预分配的文件都用 0 进行填充，由于有了这个机制，MongoDB 始终保持额外的空间和空余的数据文件，从而有效避免了由于数据暴增而带来的磁盘压力过大的问题。

由于表中数据量的增加，数据文件每新分配一次，它的大小都会是上一个数据文件大小

的2倍,每个数据文件最大2G。这样的机制有利于防止较小的数据库浪费过多的磁盘空间,同时又能保证较大的数据库有相应的预留空间使用。数据库的每张表都对应一个命名空间,每个索引也有对应的命名空间。这些命名空间的元数据都集中在*.ns文件中。

在图5-28中,foo这个数据库包含3个文件用于存储表和索引数据,foo.2文件属于预分配的空文件。foo.0和foo.1这两个数据文件被分为了相应的盘区对应不同的名字空间。

图5-28显示了命名空间和盘区的关系。每个命名空间可以包含多个不同的盘区,这些盘区并不是连续的。与数据文件的增长相同,每一个命名空间对应的盘区大小也是随着分配的次数不断增长的。这样做的目的是平衡命名空间浪费的空间与保持某一个命名空间中数据的连续性。图5-28中还有一个需要注意的命名空间:\$freelist,这个命名空间用于记录不再使用的盘区(被删除的Collection或索引)。每当命名空间需要分配新的盘区时,都会先查看\$freelist是否有大小合适的盘区可以使用。

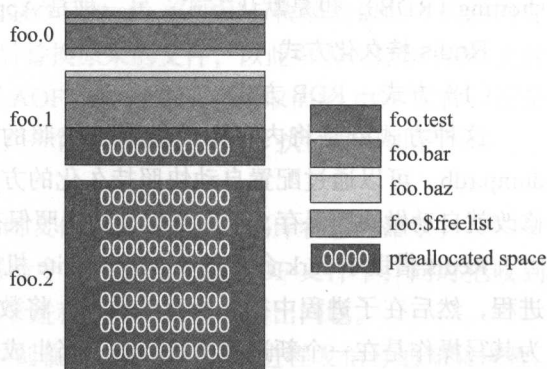


图 5-28 MongoDB 空间存储

需要注意的命名空间:\$freelist,这个命名空间用于记录不再使用的盘区(被删除的Collection或索引)。每当命名空间需要分配新的盘区时,都会先查看\$freelist是否有大小合适的盘区可以使用。

MongoDB 分布式部署方式

服务器分布式部署可以有多种方式。每台config server、mongos、mongod都可以是单独的服务器,但这样会导致某些服务器的浪费,例如config server。图5-29为物理机共享的集群部署,不需要额外加机器。

当然也有其他的方案,例如把mongos部署在所有的mongod(server1~6)上,又或者在每个运用服务器(server7)上部署mongos。这样部署有个好处在于,appserver和mongos之间的通信建立在localhost interface上,减少了通信成本。当然,此乃官方说法,但本人有想法,尽管减少了appserver和mongos之间的通信成本,但mongos与mongod之间的通信成本却增加了,而且把mongos部署在appserver上并不是很利于sa管理,MongoDB应该是一个相对独立的系统,与应用的耦合度应该尽量降到最低,万一应用想要换数据库,也能多少减少些工作量。

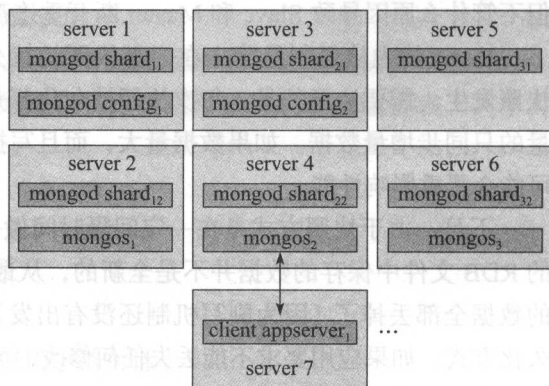


图 5-29 MongoDB 分布式部署结构

2. Redis

Redis 是一个基于 Key-Value 的高速缓存系统,类似于 memcached,但是支持更复杂的数据结构 List、Set、Sorted Set,并且有持久化的功能。支持在服务器端计算集合的并、交

和补集（difference）等，还支持多种排序功能，也可以被看成是一个数据结构服务器。由于 Redis 支持持久化的内存数据库，也就是说需要经常将内存中的数据同步到磁盘来保证持久化，以保证不会因为断电等因素丢失数据。同时，它支持两种持久化方式，一种是 Snapshotting（RDB），也是默认方式；另一种是 Append-only file（aof）的方式。

Redis 持久化方式

（1）方式一 RDB 方式

这种方式就是将内存中的数据以快照的方式写入到二进制文件中，默认的文件名为 dump.rdb。可以通过配置自动快照持久化的方式，配置 Redis 在 n 秒内如果超过 m 个 key 被修改就自动做快照保存，下面是默认的快照保存配置。

Redis 借助了 fork 命令的 copy on write 机制。在生成快照时，将当前进程 fork 出一个子进程，然后在子进程中循环所有的数据，将数据写成为 RDB 文件。RDB 文件不会坏掉，因为其写操作是在一个新进程中进行的。当生成一个新的 RDB 文件时，Redis 生成的子进程会先将数据写到一个临时文件中，然后通过原子性 rename 系统调用将临时文件重命名为 RDB 文件。这样在任何时候出现故障，Redis 的 RDB 文件都总是可用的，并且 Redis 的 RDB 文件也是 Redis 主从同步内部实现中的一环：

第一次 Slave 向 Master 同步的实现是：Slave 向 Master 发出同步请求，Master 先 dump 出 rdb 文件，然后将 rdb 文件全量传输给 Slave，然后 Master 把缓存的命令转发给 Slave，初次同步完成。

第二次以及以后的同步实现是：Master 将变量的快照直接实时依次发送给各个 Slave。但不管什么原因导致 Slave 和 Master 断开重连都会重复以上两个步骤的过程。

Redis 的主从复制是建立在内存快照的持久化基础上的，只要有 Slave 就一定会有内存快照发生。需要注意的是，每次快照持久化都是将内存数据完整写入到磁盘一次，并不是增量的只同步增量数据。如果数据量大，而且写操作比较多，必然会引起大量的磁盘 IO 操作，可能会严重影响性能。

不足：由于快照方式是在一定间隔时间做一次的，所以一旦 Redis 出现问题，那么我们的 RDB 文件中保存的数据并不是全新的，从最后一次 RDB 文件生成到 Redis 停机这段时间的数据全部丢掉了（因为刷写机制还没有出发）。RDB 就是 Snapshot 快照存储，是默认的持久化方式。如果应用要求不能丢失任何修改，可以采用 AOF 持久化方式。

（2）方式二 AOF（Append Only File）方式

AOF 方式比 RDB 方式有更好的持久化性。由于在使用 AOF 持久化方式时，Redis 会将每一个收到的写命令都通过 write 函数追加到文件中，类似于 MySQL 的 binlog。当 Redis 重启时会通过重新执行文件中保存的写命令来在内存中重建整个数据库的内容。当然由于 OS 会在内核中缓存 write 做的修改，所以可能不是立即写到磁盘上。这样 AOF 方式的持久化也还是有可能丢失部分修改。不过我们可以通过配置文件告诉 Redis 我们想要通过 fsync 函数强制 OS 写入到磁盘的时机。

AOF 的完全持久化方式同时也带来了另一个问题，持久化文件会变得越来越来大。（比如，调用 INCR test 命令 100 次，文件中就必须保存全部的 100 条命令，但其实 99 条都是多余的。因为要恢复数据库的状态其实文件中保存一条 SET test 100 就够了。）为了压缩 AOF 的持久化文件，Redis 提供了 bgrewriteaof 命令。收到此命令后 Redis 将使用与快照类似的方式将内存中的数据以命令的方式保存到临时文件中，最后替换原来的文件，以此来控制 AOF 文件的增长。由于是模拟快照的过程，因此在重写 AOF 文件时并没有读取旧的 AOF 文件，而是将整个内存中的数据库内容用命令的方式重写了一个新的 AOF 文件。执行过程如下：

- ❑ Redis 调用 fork，现在有父子两个进程。
- ❑ 子进程根据内存中的数据库快照，往临时文件中写入重建数据库状态的命令。
- ❑ 父进程继续处理 client 请求，除了把写命令写入到原来的 AOF 文件中，同时把收到的写命令缓存起来。这样就能保证如果子进程重写失败并不会出问题。
- ❑ 当子进程把快照内容写入已命令方式写到临时文件中后，子进程发信号通知父进程。然后父进程把缓存的写命令也写入到临时文件。
- ❑ 现在父进程可以使用临时文件替换老的 AOF 文件，并重命名，后面收到的写命令也开始往新的 AOF 文件中追加。

需要注意的是，重写 AOF 文件的操作，并没有读取旧的 AOF 文件，而是将整个内存中的数据库内容用命令的方式重写了一个新的 AOF 文件，这点和快照有点类似。

Redis 启动流程

整个过程如图 5-30 所示：

- ❑ 初始化 Server 变量，设置 Redis 相关的默认值。
- ❑ 读入配置文件，同时接收命令行中传入的参数，替换服务器设置的默认值。
- ❑ 初始化服务器功能模块。在这一步初始化包括进程信号处理、客户端链表、共享对象、初始化数据、初始化网络连接等。
- ❑ 从 RDB 或 AOF 重载数据。
- ❑ 网络监听服务启动前的准备工作。
- ❑ 开启事件监听，开始接受客户端的请求。

Redis 主要特点

- ❑ Redis 数据库完全在内存中，使用磁盘仅用于持久性。
- ❑ 相比许多键值数据存储，Redis 拥有一套较为丰富的数据类型。
- ❑ Redis 可以将数据复制到任意数量的从服务器。

Redis 主要优势

- ❑ 异常快速：Redis 的速度非常快，每秒能执行约 11 万集合，每秒约 81 000 条记录。
- ❑ 支持丰富的数据类型：Redis 支持最大多数开发人员已经知道像列表、集合、有序集合、散列数据类型。这使得它非常容易解决各种各样的问题，因为我们知道哪些问题是处理通过它的数据类型更好。

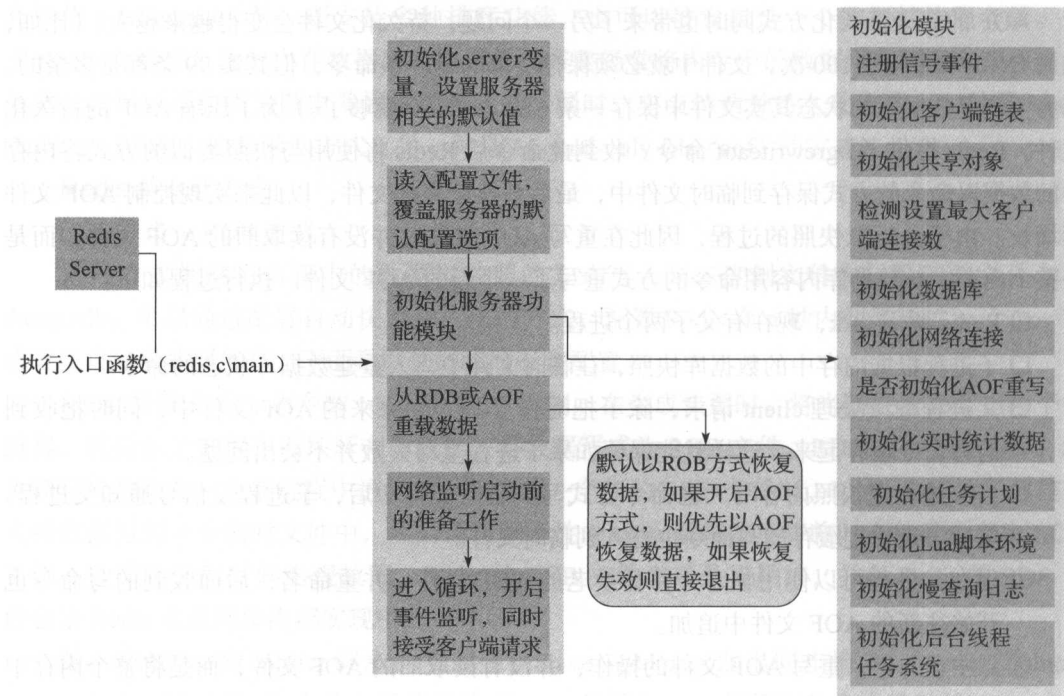


图 5-30 Redis 启动流程

- ❑ 操作都是原子性：所有的 Redis 操作都是原子性的，这保证了如果两个客户端同时访问 Redis 服务器将获得更新后的值。
- ❑ 多功能实用工具：Redis 是一个多功能实用的工具，可以在多个用例如缓存、消息、队列使用（Redis 原生支持发布 / 订阅），任何短暂的数据，应用程序如 Web 应用程序会话，网页命中计数等。

5.1.3 实时计算

1. Spark

Spark 是加州大学伯克利分校的 AMP 实验室所开源的类 Hadoop MapReduce 的通用并行框架，Spark 拥有 Hadoop MapReduce 所具有的优点；但不同于 MapReduce 的是，Job 的中间输出结果可以保存在内存中，从而不再需要读写 HDFS，因此 Spark 能更好地适用于数据挖掘与机器学习等需要迭代的 MapReduce 的算法。Spark 与 Hadoop 具有相似的开源集群计算环境，但是两者之间还存在一些不同之处，这些有用的不同之处使 Spark 在某些工作负载方面表现得更加优越。换句话说，Spark 启用了内存分布数据集，除了能够提供交互式查询外，还可以优化迭代工作负载。

Spark 是在 Scala 语言中实现的，它将 Scala 用做其应用程序框架。与 Hadoop 不同，Spark 和 Scala 能够紧密集成，其中的 Scala 可以像操作本地集合对象一样轻松地操作分布式数据

集。尽管创建 Spark 是为了支持分布式数据集上的迭代作业，但是实际上它是对 Hadoop 的补充，可以在 Hadoop 文件系统中并行运行。通过名为 Mesos 的第三方集群框架可以支持此行为。Spark 由加州大学伯克利分校 AMP 实验室 (Algorithms, Machines and People Lab) 开发，可用于构建大型的、低延迟的数据分析应用程序。

Spark 架构和基本原理

与许多专有的大数据处理平台不同，Spark 建立在统一抽象的 RDD 之上（要理解 Spark，就需要理解 RDD），使得它可以以基本一致的方式应对不同的数据处理场景，包括 Map-Reduce、Streaming、SQL、Machine Learning 以及 Graph 等。这即 Matei Zaharia 所谓的“设计一个通用的编程抽象” (Unified Programming Abstraction)。这正是 Spark 这朵小火花让人着迷的地方。如图 5-31 所示为 Spark 逻辑架构。

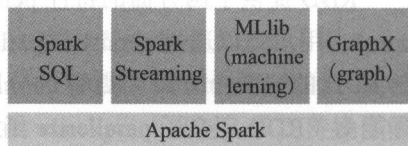


图 5-31 Spark 逻辑架构

(1) RDD 基本原理

RDD (Resilient Distributed Datasets) 是一个容错的、并行的数据结构，可以让用户显式地将数据存储到磁盘和内存中，并能控制数据的分区。同时，RDD 还提供了一组丰富的操作来操作这些数据。在这些操作中，诸如 map、flatMap、filter 等转换操作实现了 monad 模式，很好地契合了 Scala 的集合操作。除此之外，RDD 还提供了诸如 join、groupBy、reduceByKey 等更为方便的操作（注意，reduceByKey 是 action，而非 transformation），以支持常见的数据运算。

通常来讲，针对数据处理有几种常见模型，包括：Iterative Algorithms、Relational Queries、MapReduce、Stream Processing。比如，Hadoop MapReduce 采用了 MapReduces 模型，Storm 则采用了 Stream Processing 模型。RDD 混合了这四种模型，使得 Spark 可以应用于各种大数据处理场景。

RDD 作为数据结构，本质上是一个只读的分区记录集合。一个 RDD 可以包含多个分区，每个分区就是一个 dataset 片段。RDD 可以相互依赖。如果 RDD 的每个分区最多只能被一个 Child RDD 的分区使用，则称之为 narrow dependency；若多个 Child RDD 分区都可以依赖，则称之为 wide dependency。不同的操作依据其特性，可能会产生不同的依赖。比如，map 操作会产生 narrow dependency，而 join 操作则产生 wide dependency。

Spark 之所以将依赖分为 narrow 与 wide，基于两点原因：

首先，narrow dependencies 可以支持在同一个 cluster node 上以管道形式执行多条命令，例如在执行了 map 后，紧接着执行 filter。相反，wide dependencies 需要所有的父分区都是可用的，可能还需要调用类似 MapReduce 之类的操作进行跨节点传递。

其次，则是从失败恢复的角度考虑。narrow dependencies 的失败恢复更有效，因为它只需要重新计算丢失的 parent partition 即可，而且可以并行地在不同节点进行重计算。而 wide dependencies 则牵涉 RDD 各级的多个 parent partitions。图 5-32 说明了 narrow dependencies

与 wide dependencies 之间的区别。

图 5-32 来自 Matei Zaharia 撰写的论文《An Architecture for Fast and General Data Processing on Large Clusters》。图中，一个 box 代表一个 RDD，一个带阴影的矩形框代表一个 partition。

（2）RDD 如何保障数据处理效率

RDD 提供了两方面的特性 persistence 和 partitioning，用户可以通过 persist 与 partitionBy 函数来控制 RDD 的这两个方面。RDD 的分区特性与并行计算能力（RDD 定义了 parallelize 函数），使得 Spark 可以更好地利用可伸缩的硬件资源。若将分区与持久化二者结合起来，就能更加高效地处理海量数据。例如：

```
input.map(parseArticle _).partitionBy(partitioner).cache()
```

partitionBy 函数需要接受一个 Partitioner 对象，例如：

```
val partitioner = new HashPartitioner(sc.defaultParallelism)
```

RDD 本质上是一个内存数据集，在访问 RDD 时，指针只会指向与操作相关的部分。比如，存在一个面向列的数据结构，其中一个实现为 Int 的数组，另一个实现为 Float 的数组。如果只需要访问 Int 字段，RDD 的指针可以只访问 Int 数组，避免了对整个数据结构的扫描。

RDD 将操作分为两类：transformation 与 action。无论执行了多少次 transformation 操作，RDD 都不会真正执行运算，只有当 action 操作被执行时，运算才会触发。而在 RDD 的内部实现机制中，底层接口则是基于迭代器的，从而使得数据访问变得更高效率，也避免了大量中间结果对内存的消耗。

在实现时，RDD 针对 transformation 操作，都提供了对应的继承自 RDD 的类型，例如 map 操作会返回 MappedRDD，而 flatMap 则返回 FlatMappedRDD。当我们执行 map 或 flatMap 操作时，不过是将当前 RDD 对象传递给对应的 RDD 对象而已。例如：

```
def map[U: ClassTag](f: T => U): RDD[U] = new MappedRDD(this, sc.clean(f))
```

这些继承自 RDD 的类都定义了 compute 函数。该函数会在 action 操作被调用时触发，在函数内部是通过迭代器进行对应的转换操作：

```
private[spark]
class MappedRDD[U: ClassTag, T: ClassTag](prev: RDD[T], f: T => U)
  extends RDD[U](prev) {
  override def getPartitions: Array[Partition] = firstParent[T].partitions
  override def compute(split: Partition, context: TaskContext) =
    firstParent[T].iterator(split, context).map(f)
}
```

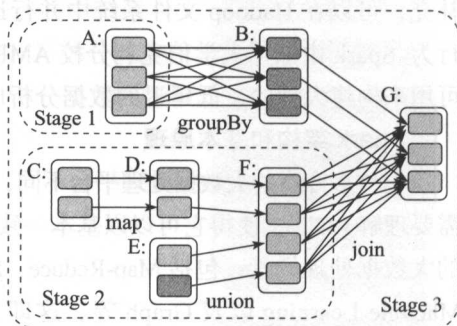


图 5-32 Spark 逻辑架构

(3) RDD 对容错的支持

支持容错通常采用两种方式：数据复制或日志记录。对于以数据为中心的系统而言，这两种方式都非常昂贵，因为它需要跨集群网络拷贝大量数据，毕竟带宽的数据远远低于内存。

RDD 天生是支持容错的。首先，它自身是一个不变的（immutable）数据集；其次，它能够记住构建它的操作图（Graph of Operation），因此当执行任务的 Worker 失败时，完全可以通过操作图获得之前执行的操作，进行重新计算。由于无需采用 replication 方式支持容错，很好地降低了跨网络的数据传输成本。

不过，在某些场景下，Spark 也需要利用记录日志的方式来支持容错。比如，在 Spark Streaming 中，针对数据进行 update 操作，或者调用 Streaming 提供的 window 操作时，就需要恢复执行过程的中间状态。此时，需要通过 Spark 提供的 checkpoint 机制，以支持操作能够从 checkpoint 得到恢复。

针对 RDD 的 wide dependency，最有效的容错方式同样还是采用 checkpoint 机制。不过，似乎 Spark 的最新版本仍然没有引入 auto checkpointing 机制。

Spark 生态圈

Spark SQL (Hive on Spark): Spark SQL 基本上就是在 Spark 的框架基础上提供和 Hive 一样的 HiveQL 命令接口，为了最大限度地保持和 Hive 的兼容性，Shark 使用了 Hive 的 API 来实现 query Parsing 和 Logic Plan generation，最后的 PhysicalPlan execution 阶段用 Spark 代替 Hadoop MapReduce。通过配置 Shark 参数，Shark 可以自动在内存中缓存特定的 RDD，实现数据重用，进而加快特定数据集的检索。同时，Shark 通过 UDF 用户自定义函数实现特定的数据分析学习算法，使得 SQL 数据查询和运算分析能结合在一起，最大化 RDD 的重复使用。

Spark Streaming: 构建在 Spark 上处理 Stream 数据的框架，基本的原理是将 Stream 数据分成小的时间片断（几秒），以类似 batch 批量处理的方式来处理这小部分数据。Spark Streaming 构建在 Spark 上，一方面是因为 Spark 的低延迟执行引擎（100ms+）可以用于实时计算；另一方面相比基于 Record 的其他处理框架（例如 Storm），RDD 数据集更容易做高效的容错处理。此外，小批量处理的方式使得它可以同时兼容批量和实时数据处理的逻辑和算法，方便了一些需要历史数据和实时数据联合分析的特定应用场合。

Bagel: Pregel on Spark，可以用 Spark 进行图计算，这是个非常有用的小项目。

Spark 计算过程

Spark 运行框架如图 5-33 所示，首先有集群资源管理服务（Cluster Manager）和运行作业任务的结点（Worker Node），然后就是每个应用的任务控制节点 Driver 和每个机器节点上有具体任务的执行进程（Executor）。

与 MR 计算框架相比，Executor 有两

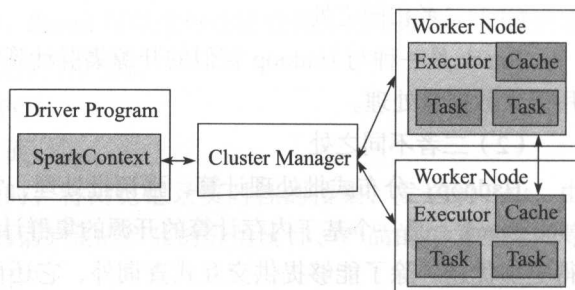


图 5-33 Spark 计算控制过程

个优点：一个是多线程来执行具体的任务，而不是像 MR 那样采用进程模型，减少了任务的启动开销；另一个是 Executor 上会有一个 BlockManager 存储模块，类似于 KV 系统（内存和磁盘共同作为存储设备），当需要迭代多轮时，可以将中间过程的数据先放到这个存储系统上，下次需要时直接读该存储系统上的数据，而不需要读写到 hdfs 等相关的文件系统里，或者在交互式查询场景下，事先将表 Cache 到该存储系统上，提高读写 IO 性能。另外 Spark 在做 Shuffle 时，在 Groupby、Join 等场景下去掉了不必要的 Sort 操作，相比于 MapReduce 只有 Map 和 Reduce 两种模式，Spark 还提供了更加丰富全面的运算操作如 filter、groupby、join 等。

Spark 采用了 Scala 来编写，在函数表达上 Scala 有天然的优势，因此在表达复杂的机器学习算法能力比其他语言更强且简单易懂。提供各种操作函数来建立起 RDD 的 DAG 计算模型。把每一个操作都看成构建一个 RDD 来对待，而 RDD 则表示的是分布在多台机器上的数据集，并且可以带上各种操作函数（如图 5-34 所示）。

```
Ex: errors = textFile ( "log" ) . filter ( _ . contains ( "error" ) )
      . map ( _ . split ( ' \t ' ) ( 1 ) )
      . cache ( )
```

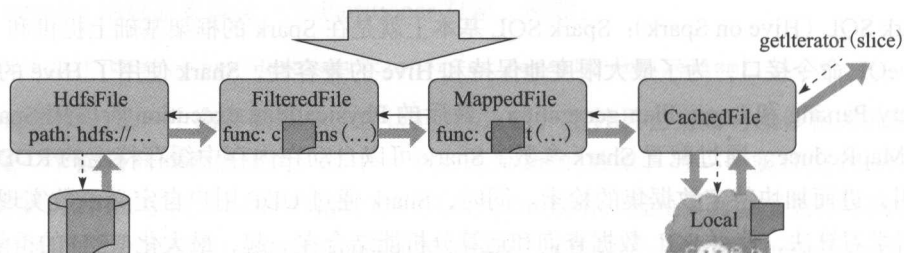


图 5-34 Spark 函数模型

首先从 hdfs 文件里读取文本内容构建成一个 RDD，然后使用 filter() 操作来对上次的 RDD 进行过滤，再使用 map() 操作取得记录的第一个字段，最后将其 Cache 在内存上，后面就可以对之前 Cache 过的数据做其他的操作。整个过程都将形成一个 DAG 计算图，每个操作步骤都有容错机制，同时还可以将需要多次使用的数据 Cache 起来，供后续迭代使用。

Spark 与 Hadoop 的异同

(1) 二者相同之处

Spark 是一种与 Hadoop 相似的开源集群计算环境，都能实现分布式数据计算和处理，可用于大数据量处理。

(2) 二者不同之处

Hadoop：分布式批处理计算，强调批处理，常用于数据挖掘、分析。

Spark：是一个基于内存计算的开源的集群计算系统，使 Spark 在某些工作负载方面表现得更优越，除了能够提供交互式查询外，它还可以优化迭代工作负载，目的是让数据分析更加快速。

- Spark 是在 Scala 语言中实现的，它将 Scala 用做其应用程序框架。Spark 和 Scala 能够紧密集成，其中的 Scala 可以像操作本地集合对象一样轻松地操作分布式数据集。
- 尽管创建 Spark 是为了支持分布式数据集上的迭代作业，但是实际上它是对 Hadoop 的补充，可以在 Hadoop 文件系统中并行运行。通过名为 Mesos 的第三方集群框架可以支持此行为。
- Spark 提供了具有有用差异的一个新的集群计算框架。Spark 是为集群计算中的特定类型的工作负载而设计，即那些在并行操作之间重用工作数据集（例如机器学习算法）的工作负载。为了优化这些类型的工作负载，Spark 引进了内存集群计算的概念，可在内存集群计算中将数据集缓存在内存中，以缩短访问延迟。
- 在大数据处理方面相信大家对于 Hadoop 已经耳熟能详，基于 GoogleMap/Reduce 来实现的 Hadoop 为开发者提供了 map、reduce 语言，使并行批处理程序变得非常简单和优美。Spark 提供的数据集操作类型有很多种，不像 Hadoop 只提供了 Map 和 Reduce 两种操作。比如，map、filter、flatMap、sample、groupByKey、reduceByKey、union、join、cogroup、mapValues、sort、partitionBy 等多种操作类型，他们把这些操作称为 Transformations。另外，Spark 同时还提供 Count、collect、reduce、lookup、save 等多种 actions。这些多种多样的数据集操作类型，给上层应用者提供了方便。各个处理节点之间的通信模型不再像 Hadoop 那样就是唯一的 Data Shuffle 一种模式。用户可以命名、物化、控制中间结果的分区等。可以说编程模型比 Hadoop 更灵活。

2. Storm

Storm 是 Twitter 开源的一个类似于 Hadoop 的实时数据处理框架，它原来是由 BackType 开发，后 BackType 被 Twitter 收购，将 Storm 作为 Twitter 的实时数据分析系统。实时数据处理的应用场景很广泛，例如商品推荐、广告投放，它可以根据当前情景上下文（用户偏好、地理位置、已发生的查询和点击等）来估计用户点击的可能性并实时做出调整。同时，Twitter 列举了 Storm 的三大作用领域：

- 信息流处理（Stream Processing）：Storm 可以用来实时处理新数据和更新数据库，兼具容错性和可扩展性，它可以用来处理源源不断的消息，并将处理之后的结果保存到持久化介质中。
- 连续计算（Continuous Computation）：Storm 可以进行连续查询并把结果即时反馈给客户，例如将 Twitter 上的热门话题发送到客户端。
- 分布式远程过程调用（Distributed RPC）。

除此之外，Storm 也被广泛用于以下方面：

- 精确的广告推送：在用户浏览产品时，将浏览记录实时性的搜集，发送到 Bolt，由 Bolt 根据用户的账户信息（如果有的话）完成产品的分类统计、产品的相关性查询等逻辑计算之后，将计算结果推送给用户。
- 实时日志的处理：Storm 可以和一个分布式存储结合起来，实时性地从多个数据源发

送数据到处理逻辑 Bolts，Bolts 完成一些逻辑处理之后，交给分布式存储框架进行存储，此时，Spout 可以是多个。

□ 高并发查询：Storm 可以用来并行处理密集查询，Storm 的拓扑结构是一个等待调用信息的分布函数，当它收到一条调用信息后，会对查询进行计算，并返回查询结果。

Storm 集群的基本组件

Storm 是一个分布式、高容错的实时计算系统，Storm 对于实时计算的意义相当于 Hadoop 对于批处理的意义。Hadoop 提供了 Map 和 Reduce 原语，使得对数据进行批处理变得非常简单和优美。同样，Storm 也对数据的实时计算提供了简单 Spout 和 Bolt 原语。Storm 集群表面上看和 Hadoop 集群非常像，但是在 Hadoop 上运行的是 MapReduce 的 Job，而在 Storm 上运行的是 Topology（拓扑），它们是非常不一样的，关键的区别是：一个 MapReduce Job 最终会结束，而一个 Topology 永远运行（除非显式地杀掉它）。

Storm 集群中有两种节点：控制节点（Master Node）和工作节点（Worker Node），控制节点上运行一个后台程序：Nimbus，它的作用类似 Hadoop 中的 JobTracker。Nimbus 负责在集群中分布代码，分配工作给机器，并且监控状态。每一个工作节点上运行一个叫做 Supervisor 的节点（类似 TaskTracker）。Supervisor 会监听分配给它那台机器的工作，根据需要启动/关闭 Worker 工作进程。每一个工作进程执行一个 Topology（类似 Job）的一个子集；一个运行的 Topology 由运行在很多机器上的多个工作进程 Worker（类似 Child）组成。

Nimbus 和 Supervisor 之间的所有协调工作都是通过一个 Zookeeper 集群来完成的，并且 Nimbus 进程和 Supervisor 都是快速失败（fail-fast）和无状态的，所有的状态要么在 Zookeeper 中，要么在本地磁盘上。这也就意味着可以用 kill-9 来杀死 Nimbus 和 Supervisor 进程，然后再重启它们，它们可以继续工作，就好像什么都没有发生过似的，这个设计使得 Storm 不可思议的稳定。逻辑结构如图 5-35 所示。

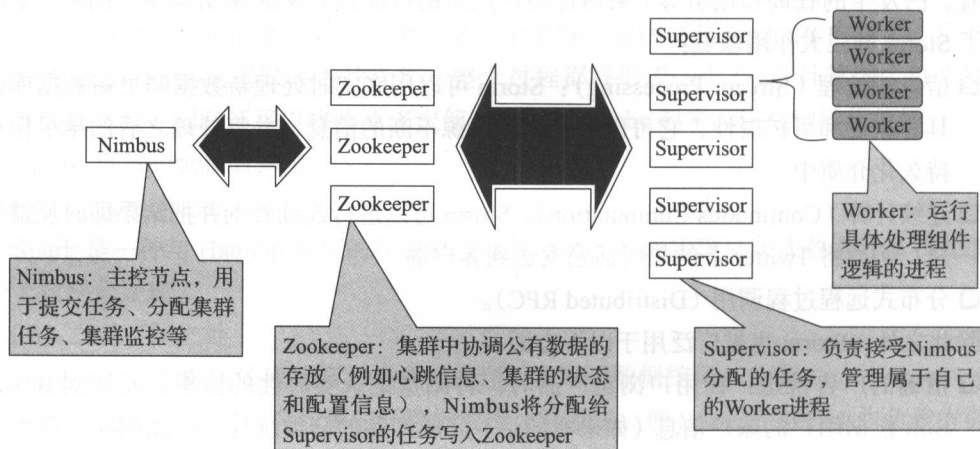


图 5-35 Storm topology 结构

(1) Topologies——作业拓扑

为了在 Storm 上做实时计算，要去建立一些 topologies。一个 topology 就是一个计算节点所组成的图。Topology 中的每个处理节点都包含处理逻辑，而节点之间的连接则表示数据流动的方向。运行一个 topology 是很简单的。首先，把你所有的代码以及所依赖的 jar 打进一个 jar 包。然后运行类似下面的这个命令：

```
strom jar all-your-code.jar backtype.Storm.MyTopology arg1 arg2
```

这个命令会运行主类：backtype.storm.MyTopology，参数是 arg1 和 arg2。这个类的 main 函数定义这个 topology 并且把它提交给 Nimbus。Storm jar 负责连接到 Nimbus 并且上传 jar 文件。因为 topology 的定义其实就是一个 Thrift 结构并且 Nimbus 就是一个 Thrift 服务，可以用任何语言创建并且提交 topology。上面的方法是用 JVM-based 语言提交的最简单的方法。Spout 和 Bolt 所组成的一个网络会被打包成 topology，topology 是 Storm 中最高一级的抽象（类似 Job）。



提示

Thrift 是一个软件框架，用来进行可扩展且跨语言的服务的开发。它结合了功能强大的软件堆栈和代码生成引擎，以构建在 C++、Java、Python、PHP、Ruby、Erlang、Perl、Haskell、C#、Cocoa、JavaScript、Node.js、Smalltalk 和 OCaml 这些编程语言间无缝结合的、高效的服务。

(2) Stream——数据流

Stream 是 Storm 中的关键抽象，一个 Stream 是一个没有边界的 Tuple 序列。Storm 提供一些原语来分布式地、可靠地把一个 Stream 传输进一个新的 Stream。比如，你可以把一个 Tweets 流传输到热门话题的流。Storm 提供的最基本的处理 Stream 的原语是 Spout 和 Bolt，可以实现 Spout 和 Bolt 对应的接口以处理你应用的逻辑。

Spout——流的源头：比如，一个 Spout 可能从 Kestrel 队列中读取消息并且把这些消息发射成一个流。又如，一个 Spout 可以调用 Twitter 的一个 API 并且把返回的 Tweets 发射成一个流。通常 Spout 会从外部数据源（队列、数据库等）读取数据，然后封装成 Tuple 形式，之后发送到 Stream 中。Spout 是一个主动的角色，在接口内部有个 nextTuple 函数，Storm 框架会不停地调用该函数。

(3) Bolt——处理逻辑

Bolt 负责处理输入的 Stream，并产生新的输出 Stream。Bolt 可以执行过滤、函数操作、Join、操作数据库等任何操作。Bolt 是一个被动的角色，其接口中有一个 Execute (Tuple input) 方法，在接收到消息之后会调用此函数，用户可以在此方法中执行自己的处理逻辑。Bolt 可以接收任意多个输入 Stream，做一些处理，有些 Bolt 可能还会发射一些新的 Stream。一些复杂的流转换，例如从一些 Tweet 中计算出热门话题，需要多个步骤，从而也

就需要多个 Bolt。Bolt 可以做任何事情：运行函数、过滤 Tuple、聚合、合并以及访问数据库等。

注意：

□ Topology 中的每一个节点都是并行运行的。在 topology 中，可以指定每个节点的并行度，Storm 则会在集群中分配多个线程来同时计算。

□ 一个 topology 会一直运行直到显式停止它。Storm 自动重新分配一些运行失败的任务，并且 Storm 保证你不会有数据丢失，即使在一些机器意外停机并且消息被丢掉的情况下。

运行中的 topology 主要由以下三个组件组成：Worker processes、Executors threads 以及 Tasks，如图 5-36 所示。

它们的数量关系如图 5-37 和图 5-38 所示。

Spout 或者 Bolt 的 Task 个数一旦指定之后就不能改变了，而 Executor 的数量可以根据情况来进行动态的调整。默认情况下 $\# \text{executor} = \# \text{tasks}$ 即一个 Executor 中运行着一个 Task。

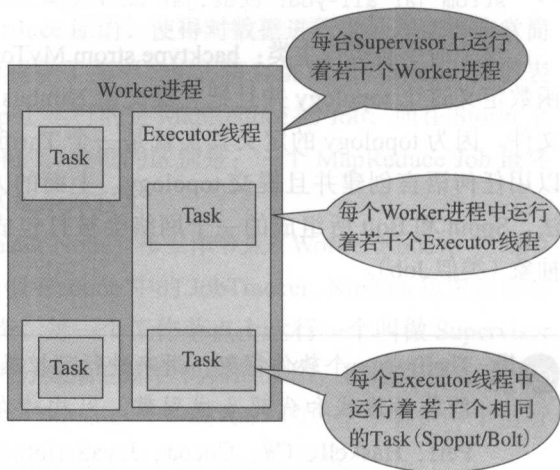


图 5-36 Topology 的组成

```
Config conf = new Config();
//设置worker数
conf.setNumWorkers(2);

//设置Executor数量

topologyBuilder.setSpout("blue-spout", new BlueSpout(), 2);
topologyBuilder.setBolt("green-bolt", new GreenBolt(), 2)
    .setNumTasks(4) //设置Task数量
    .shuffleGrouping("blue-spout");

topologyBuilder.setBolt("yellow-bolt", new YellowBolt(), 6)
    .shuffleGrouping("green-bolt");
```

图 5-37 Executor 和 Tasks 的数量关系（一）

数据模型

Storm 使用 Tuple 来作为它的数据模型。每个 Tuple 是一堆值，每个值有一个名字，并且每个值可以是任何类型，一个 Tuple 可以看做一个没有方法的 Java 对象。总体来看，Storm 支持所有的基本类型、字符串以及字节数组作为 Tuple 的值类型。也可以使用自己定义的类型来作为值类型，只要实现对应的序列化器 (Serializer)。

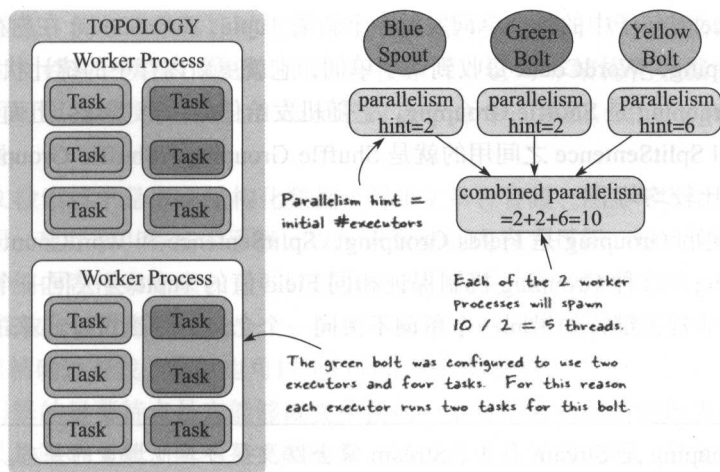


图 5-38 Executor 和 Tasks 的数量关系 (二)

一个 Tuple 代表数据流中一个基本的处理单元，例如一条 cookie 日志，它可以包含多个 Field，每个 Field 表示一个属性。

一个没有边界的、源源不断的、连续的 Tuple 序列就组成了 Stream。Tuple 本来应该是一个 Key-Value 的 Map，由于各个组件间传递的 Tuple 的字段名称已经事先定义好了，所以 Tuple 只需要按序填入各个 Value，所以就是一个 Value List。

流分组策略 (Stream Grouping)

流分组策略告诉 topology 如何在两个组件之间发送 Tuple。要记住，Spouts 和 Bolts 以很多 Task 的形式在 topology 中同步执行。如果从 Task 的粒度来看一个运行的 topology，它应该是图 5-39 这样的。

从 Task 角度来看 topology：

当 Bolt A 的一个 Task 要发送一个 Tuple 给 Bolt B，它应该发送给 Bolt B 的哪个 Task 呢？

Stream Grouping 专门回答这种问题。在我们深入研究不同的 Stream Grouping 之前，让我们看一下 Storm-starter 中的另外一个 topology。Word CountTopology 读取一些句子，输出句子中每个单词出现的次数。

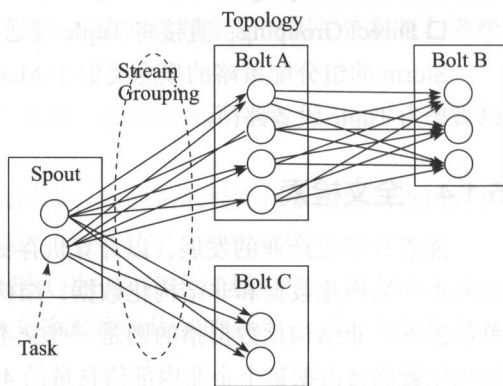


图 5-39 Storm 中各个对象的示意图

```
TopologyBuilder builder = new TopologyBuilder();
builder.setSpout( , new RandomSentenceSpout(), );
builder.setBolt( , new SplitSentence(), )
    .shuffleGrouping( );
builder.setBolt( , new WordCount(), )
    .fieldsGrouping( , new Fields( "word" ) );
```

SplitSentence 对句子中的每个单词发射一个新的 Tuple，WordCount 在内存中维护一个单词→次数的 mapping，WordCount 每收到一个单词，它就更新内存中的统计状态。

最简单的 grouping 是 Shuffle Grouping，它随机发给任何一个 Task。上面例子中 Random SentenceSpout 和 SplitSentence 之间用的就是 Shuffle Grouping，Shuffle Grouping 对各个 Task 的 Tuple 分配得比较均匀。

一种更有趣的 Grouping 是 Fields Grouping，SplitSentence 和 WordCount 之间使用的就是 Fields Grouping，这种 Grouping 机制保证相同 Field 值的 Tuple 会去同一个 Task，这对于 WordCount 来说非常关键，如果同一个单词不去同一个 Task，那么统计出来的单词次数就不对了。



注意 fields grouping 是 Stream 合并、Stream 聚合以及很多其他场景的基础。在背后，fields grouping 使用一致性哈希来分配 Tuple。

Storm 支持的组分配策略如下：

- ❑ Shuffle Grouping：随机选择一个 Task 来发送。
- ❑ Fields Grouping：根据 Tuple 中的 Fields 来做一致性 hash，相同 hash 值的 Tuple 被发送到相同的 Task。
- ❑ All Grouping：广播发送，将每一个 Tuple 发送到所有的 Task。
- ❑ Global Grouping：所有的 Tuple 会被发送到某个 Bolt 中的 id 最小的那个 Task。
- ❑ None Grouping：不关心 Tuple 发送给哪个 Task 来处理，等价于 ShuffleGrouping。
- ❑ Direct Grouping：直接将 Tuple 发送到指定的 Task 来处理。

Storm 的组分配策略的概念类似于 MapReduce 的 Partition 机制，通过使用一些分组策略原语来为 Tuple 设置路由。

5.1.4 全文检索

随着计算机产业的发展，以计算机存储设备为载体的数据愈来愈多，这些数据大致可分为两类：结构化数据和非结构化数据，结构化数据指的是诸如企业财务账目和生产数据、订单数据等，非结构化数据指的则是一些文本数据、图像声音等多媒体数据等，一般情况下非结构化数据会占据整个企业内部信息量的 40% 以上。

那么什么叫做全文检索呢？这就要从我们生活中的数据说起。我们生活中的数据总体分为两种：结构化数据和非结构化数据。结构化数据指具有固定格式或有限长度的数据，例如数据库、元数据等。非结构化数据指不定长或无固定格式的数据，例如邮件、Word 文档等。当然有的地方还会提到第三种，半结构化数据，例如 XML、HTML 等，根据需要可按结构化数据来处理，也可抽取纯文本按非结构化数据来处理。

而非结构化数据的另一种叫法叫全文数据。按照数据的分类，搜索也分为两种：①对

结构化数据的搜索,例如对数据库的搜索,用 SQL 语句。再如,对元数据的搜索,如利用 Windows 搜索对文件名、类型、修改时间进行搜索等。②对非结构化数据的搜索,如利用 Windows 搜索也可以搜索文件内容, Linux 下的 grep 命令;再如,用 Google 和百度可以搜索大量内容数据。

所以,全文检索技术是以非结构化数据(例如文章、音频、图片等)为主要内容,通过文本解析、分词、建立索引等技术手段,以检索数据的内容为目标,输入文章中任意一段信息将所需数据详细内容检索出来。它可以根据全文中有关片断(标题、章、节、段、句、词等信息),快速获取出该片断所在的全文,类似于给整本书的每个字词添加一个标签并建立目录,使之能快速精确地查找,同时也可以进行各种统计和分析。

全文搜索引擎的显著特点是它能够以文中任何一个有检索意义的词作为检索入口,而且取得的检索结果是原始文献,而不是文献线索。经过近十几年的发展,全文检索从最初的字符串匹配程序已经演进到能对超大文本、语音、图像、活动影像等非结构化数据进行综合管理的大型软件。由于内涵和外延的深刻变化,全文检索系统已成为新一代管理信息系统的代名词,衡量全文检索系统的基本指标也逐渐形成规范。

搜索引擎是全文检索技术最主要的一个应用,起源于传统的信息全文检索理论,即计算机程序通过扫描每一篇文章中的每一个词,建立以词为单位的倒排文件,检索程序根据检索词在每一篇文章中出现的频率和每一个检索词在一篇文章中出现的概率,对包含这些检索词的文章进行排序,最后输出排序的结果。全文检索技术是搜索引擎的核心支撑技术。

一个好的搜索引擎是一个理想站点的关键。很多人在访问一个站点时喜欢使用站点检索,站点检索应是分类目录导航和全文检索的完美结合,具体包括以下几个方面:

- 首先,我们关注的是查全率,即系统在进行某一检索时,检索出的相关资料量与系统资料库中相关资料总量的比率。
- 查准率则是保证我们找到最有用资料的一个关键,是系统在进行某一检索时,检索出的有用资料数量与检索出资料总量的比率。
- 检索速度或者说响应时间是提高工作效率的保障,指的是从提交检索课题到查出资料结果所需的时间。最基本的检索速度是应该达“千万汉字,秒级响应”。

还有诸如收录范围(所查找的范围)、用户负担(用户在检索过程中付出精力的总和)、输出形式(输出信息表现形式)等指标也是衡量全文检索系统优劣的要素。下面详细介绍目前最常用的 3 款企业级全文检索引擎。

1. Lucene

Lucene 是一个高效的,基于 Java 的全文检索库。最初是由技术大神 Doug Cutting(Hadoop 项目发起人)开发的,2001 年作为高质量的开源 Java 产品加入到 Apache 软件基金会家族中,2002 年 6 月 Lucene1.2 版正式在 Apache 发布。作为一个开放源代码项目, Lucene 从问世之后,引发了开放源代码社群的巨大反响,程序员不仅使用它构建具体的全文检索应用,而且将之集成到各种系统软件中,以及构建 Web 应用,甚至一些商业软件也采用它作为其内部全

文检索子系统的核心。Lucene 以其开放源代码的特性、优异的索引结构、良好的系统架构获得了越来越多的应用。随着每个版本的发布，这个项目得到明显的增强，也吸引了更多的用户和开发人员，至今已经是 5.0 以上。

对非结构化数据即对全文数据的搜索主要有两种方法：一种是顺序扫描法（Serial Scanning）。所谓顺序扫描，例如要找内容包含某一个字符串的文件，就是每一个文档都逐一查看，对于每一个文档，从头看到尾，如果此文档包含此字符串，则此文档为我们要找的文件，接着看下一个文件，直到扫描完所有的文件。比如，利用 Windows 搜索也可以搜索文件内容，只是相当慢。如果你有一个 80G 的硬盘，且想在其中找到一个内容包含某字符串的文件，不花几个小时，怕是做不到。Linux 下的 `grep` 命令也是这一种方式。大家可能觉得这种方法比较原始，但对于小数据量的文件，这种方法还是最直接、最方便的。但是对于大量的文件，这种方法就很慢了。

有人可能会说，对非结构化数据顺序扫描很慢，对结构化数据的搜索却相对较快（由于结构化数据有一定的结构可以采取一定的搜索算法加快速度），那么把我们的非结构化数据想办法弄得有一定结构不就行了吗？

这种想法很自然，却构成了全文检索的基本思路，即将非结构化数据中的一部分信息提取出来，重新组织，使其变得有一定结构，然后对此有一定结构的数据进行搜索，从而达到搜索相对较快的目的。

这部分从非结构化数据中提取出的然后重新组织的信息，从而形成第二种搜索方法，我们称之为索引。

这种说法比较抽象，举几个例子就很容易明白，例如字典，字典的拼音表和部首检字表就相当于字典的索引，对每一个字的解释是非结构化的，如果字典没有音节表和部首检字表，在茫茫辞海中找一个字只能顺序扫描。然而字的某些信息可以提取出来进行结构化处理，例如读音就比较结构化，分声母和韵母，分别只有几种可以一一列举，于是将读音拿出来按一定的顺序排列，每一项读音都指向此字的详细解释的页数。我们搜索时按结构化的拼音搜到读音，然后按其指向的页数，便可找到我们的非结构化数据——即对字的解释。

Lucene 就是建立索引，再对索引进行搜索的过程的技术实现。图 5-40 来自“Lucene in action”，不仅仅描述了 Lucene 的检索过程，也描述了全文检索的一般过程。

使用 Lucene 进行全文检索大体分两个过程，索引创建（Indexing）和搜索索引（Search）。

□ 索引创建：对现实世界中所有的结构化和非结构化数据提取信息、创建索引的过程。

□ 搜索索引：就是得到用户的查询请求，搜索创建的索引，然后返回结果的过程。

Lucene 中最常用的 4 个 Jar：

□ 首先是最最重要的一个，`Lucene-core-4.0.0.jar`，其中包括常用的文档、索引、搜索、存储等相关核心代码。

□ `Lucene-analyzers-common-4.0.0.jar`，其中包含各种语言的词法分析器，用于对文件内容进行关键字切分、提取。

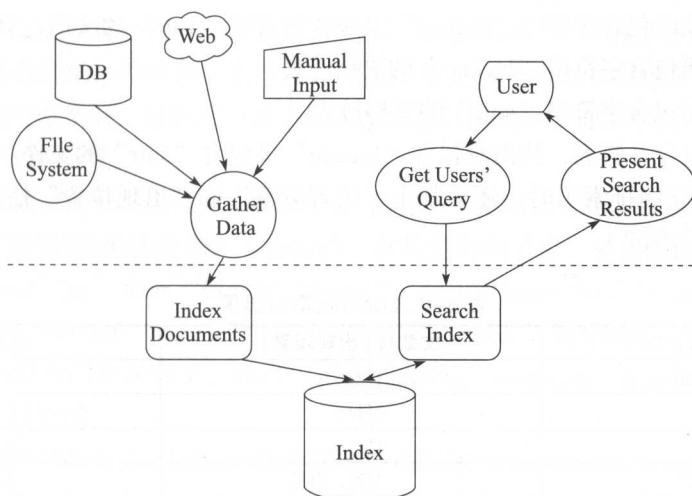


图 5-40 Lucene 检索过程

- ❑ Lucene-highlighter-4.0.0.jar，该 jar 包主要用于将搜索出的内容高亮显示。
- ❑ Lucene-queryparser-4.0.0.jar，该 jar 包提供了搜索相关的代码，用于各种搜索，例如模糊搜索、范围搜索等。

索引中究竟存些什么

首先我们来看为什么顺序扫描的速度慢：其实是由于我们想要搜索的信息和非结构化数据中所存储的信息不一致造成的。非结构化数据中所存储的信息是每个文件包含哪些字符串，即已知文件，欲求字符串相对容易，即是从文件到字符串的映射。而我们想搜索的信息是哪些文件包含此字符串，即已知字符串，欲求文件，即是从字符串到文件的映射。两者恰恰相反。于是如果索引总能够保存从字符串到文件的映射，则会大大提高搜索速度。

由于从字符串到文件的映射是文件到字符串映射的反向过程，于是将保存这种信息的索引称为倒排索引（也称为反向索引）。倒排索引所保存的信息一般如下：假设我的文档集合中有 100 篇文档，为了方便表示，我们为文档编号从 1 到 100，得到图 5-41 的结构（示例 1）。

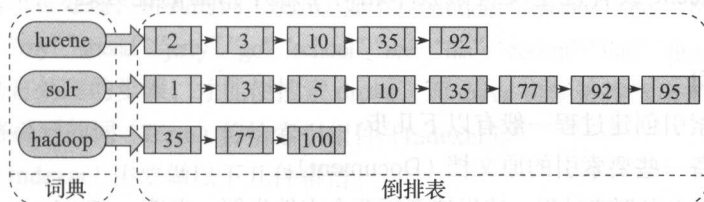


图 5-41 Lucene 索引结构

左边保存的是一系列字符串，称为词典。每个字符串都指向包含此字符串的文档（Document）链表，此文档链表称为倒排表（Posting List）。有了索引，便使保存的信息和要搜索的信息一致，可以大大加快搜索的速度。

比如，要寻找既包含字符串“lucene”又包含字符串“solr”的文档，只需要以下几步：

第一步：取出包含字符串“lucene”的文档链表。

第二步：取出包含字符串“solr”的文档链表。

第三步：通过合并链表，找出既包含“lucene”又包含“solr”的文件。

当 Lucene 建立倒排索引时，还会加上“出现频率”和“出现位置”信息，我们的索引结构变为表 5-4（示例 2）。

表 5-4 Lucene 索引结果

关键词	文章号 [出现频率]	出现位置
guangzhou	1[2]	3, 6
he	2[1]	1
i	1[1]	4
live	1[2], 2[1]	2, 5, 2
shanghai	2[1]	3
tom	1[1]	1

以 live 为例我们说明一下该结构：live 在文章 1 中出现了 2 次，文章 2 中出现了一次，它的出现位置为“2, 5, 2”，这表示什么呢？我们需要结合文章号和出现频率来分析，文章 1 中出现了 2 次，那么“2, 5”就表示 live 在文章 1 中出现的两个位置，文章 2 中出现了一次，剩下的“2”就表示 live 是文章 2 中的第 2 个关键字。

以上就是 Lucene 索引结构中最核心的部分。我们注意到关键字是按字符顺序排列的（Lucene 没有使用 B 树结构），因此 Lucene 可以用二元搜索算法快速定位关键词。

看到这个地方，有人可能会说，该方法的确加快了搜索的速度，但是多了索引的过程，两者加起来不一定比顺序扫描快多少。的确，加上索引的过程，全文检索不一定比顺序扫描快，尤其是在数据量小时更是如此。而对大量数据创建索引也是一个很慢的过程。

然而两者还是有区别的，顺序扫描是每次都要扫描，而创建索引的过程仅仅需要一次，以后便是一劳永逸了，每次搜索，创建索引的过程不必经过，仅仅搜索创建好的索引即可。这也是 Lucene 或者说全文检索技术相对于顺序扫描的优势之一：一次索引，多次使用。

如何创建索引

全文检索的索引创建过程一般有以下几步：

第一步：准备一些要索引的原文档（Document）。

为了方便说明索引创建过程，这里特意用两个文件为例，文件一：Students should be allowed to go out with their friends, but not allowed to drink beer。文件二：My friend Jerry went to school to see his students but found them drunk which is not allowed。

第二步：将原文档传给分词组件（Tokenizer）。

分词组件（Tokenizer）会做以下几件事情（此过程称为 Tokenize）：①将文档分成一个一

个单独的单词。②去除标点符号。③去除停用词 (Stop Word)。所谓停用词 (Stop Word) 就是一种语言中最普通的一些单词, 由于没有特别的意义, 因而大多数情况下不能成为搜索的关键词, 因而创建索引时, 这种词会被去掉而减少索引的大小。英语中的停用词 (Stop Word) 如: “the” “a” “this” 等。

对于每一种语言的分词组件 (Tokenizer), 都有一个停用词 (Stop Word) 集合。经过分词 (Tokenizer) 后得到的结果称为词元 (Token)。在我们的例子中, 便得到以下词元 (Token): “Students” “allowed” “go” “their” “friends” “allowed” “drink” “beer” “My” “friend” “Jerry” “went” “school” “see” “his” “students” “found” “them” “drunk” “allowed”。这里面使用的是 Lucene 标准的词法分析器, 如果专门针对中文, 还可以搭配 paoding、mmseg4j、ikanalyzer 等主流第三方中文分词组件进行使用。

第三步: 将得到的词元 (Token) 传给语言处理组件 (Linguistic Processor)。

语言处理组件 (Linguistic Processor) 主要是对得到的词元 (Token) 做一些同语言相关的处理。对于英语, 语言处理组件 (Linguistic Processor) 一般做以下几点:

❑ 变为小写 (Lowercase)。

❑ 将单词缩减为词根形式, 如 “cars” 到 “car” 等。这种操作称为: stemming。

❑ 将单词转变为词根形式, 如 “drove” 到 “drive” 等。这种操作称为: lemmatization。

stemming 和 lemmatization 的相同之处为两者都要使词汇成为词根形式。不同之处为 stemming 采用的是“缩减”的方式: “cars” 到 “car”, “driving” 到 “drive”。而 lemmatization 采用的是“转变”的方式: “drove” 到 “drive”, “driving” 到 “drive”。同时两者的算法也有不同, stemming 主要是采取某种固定的算法来做这种缩减, 如去除 “s”, 去除 “ing” 加 “e”, 将 “ational” 变为 “ate”, 将 “tional” 变为 “tion”。lemmatization 主要是采用保存某种字典的方式做这种转变。比如, 字典中有 “driving” 到 “drive”, “drove” 到 “drive”, “am, is, are” 到 “be” 的映射, 做转变时, 只要查字典即可。

同时, stemming 和 lemmatization 不是互斥关系, 是有交集的, 有的词利用这两种方式都能达到相同的转换。语言处理组件 (Linguistic Processor) 的结果称为词 (Term)。在我们的例子中, 经过语言处理, 得到的词 (Term) 如下: “student” “allow” “go” “their” “friend” “allow” “drink” “beer” “my” “friend” “jerry” “go” “school” “see” “his” “student” “find” “them” “drink” “allow”。也正是因为有语言处理的步骤, 才能使搜索 drove, 而 drive 也能被搜索出来。

第四步: 将得到的词 (Term) 传给索引组件 (Indexer)。

索引组件 (Indexer) 主要做以下几件事情:

利用得到的词 (Term) 创建一个字典。在我们的例子中, 字典如表 5-5 所示。

表 5-5 字典示例

Term	Document ID	Term	Document ID
student	1	allow	1



(续)

Term	Document ID	Term	Document ID
go	1	go	2
their	1	school	2
friend	1	see	2
allow	1	his	2
drink	1	student	2
beer	1	find	2
my	2	them	2
friend	2	drink	2
jerry	2	allow	2

对字典按字母顺序进行排序，如表 5-6 所示。

表 5-6 字典排序后示例

Term	Document ID	Term	Document ID
allow	1	go	2
allow	1	his	2
allow	2	jerry	2
beer	1	my	2
drink	1	school	2
drink	2	see	2
find	2	student	1
friend	1	student	2
friend	2	their	1
go	1	them	2

合并相同的词 (Term) 成为文档倒排 (Posting List) 链表 (如图 5-42 所示)。

在此表中，有几个定义：Document Frequency 即文档频次，表示总共有多少文件包含此词 (Term)。Frequency 即词频率，表示此文件中包含了几个此词 (Term)。所以对词 (Term) “allow” 来讲，总共有两篇文档包含此词 (Term)，从而词 (Term) 后面的文档链表总共有两项，第一项表示包含 “allow” 的第一篇文档，即 1 号文档，此文档中，“allow” 出现了两次，第二项表示包含 “allow” 的第二个文档，是 2 号文档，此文档中，“allow” 出现了一次。

Lucene 创建索引的具体实现如下：

第一步：确定一个词法分析器。

```
Analyzer analyzer = new StandardAnalyzer(Version.LUCENE_CURRENT);# 参数中的 Version.LUCENE_CURRENT, 代表使用当前的 Lucene 版本, 本文环境中也可以写成 Version.LUCENE_40.
```



```
String text2 = " My friend Jerry went to school to see his students but found
them drunk which is not allowed.";
doc.add(new Field("fieldname", text2, TextField.TYPE_STORED));
iwriter.addDocument (doc);
iwriter.close();
```

到此为止，索引已经创建好了，我们可以通过它很快地我们想要的文档。而且在此过程中，我们惊喜地发现，搜索“drive”“driving”“drove”“driven”也能够被搜到。因为在我们的索引中，“driving”“drove”“driven”都会经过语言处理而变成“drive”，在搜索时，如果您输入“driving”，输入的查询语句同样经过我们这里的第一到三步，从而变为查询“drive”，进而可以搜索到想要的文档。

搜索及结果排序原理

索引创建存储并可以提取了之后，事情其实并没有结束，找到了仅仅是一个方面。如果仅仅只有 1 个或 10 个文档包含我们查询的字符串，我们的确找到了。然而如果结果有 1000 个，甚至成千上万个呢？哪个又是最想要的文件呢？

打开百度，例如你想了解关于 CPU 的介绍，于是输入“CPU”，却发现总共有 100 000 000 多个结果返回，如图 5-43 所示。这么庞大的答案，突然发现找不到是一个问题，找到的太多也是一个问题。在如此多的结果中，如何将最相关的放在最前面呢？

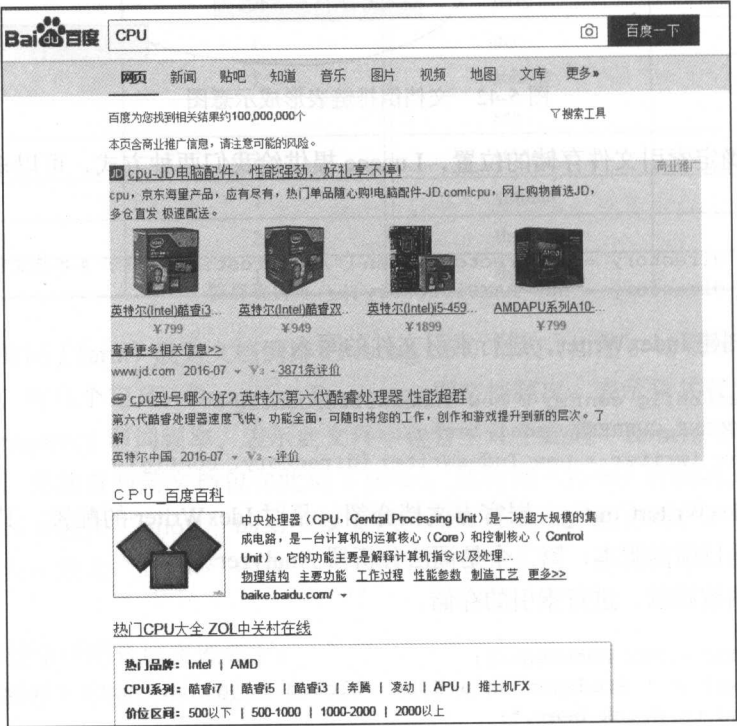


图 5-43 百度搜索 CPU 后的返回结果

如何在成千上万的搜索结果中，找到和查询语句最相关的呢？如何判断搜索出的文档和查询语句的相关性呢？这要回到我们的三个问题：如何对索引进行搜索，搜索主要分为以下几步：

第一步：用户输入查询语句。

查询语句同我们普通的语言一样，也是有一定语法的。不同的查询语句有不同的语法，例如 SQL 语句就有一定的语法。查询语句的语法根据全文检索系统的实现而不同，最基本的有：AND、OR、NOT 等。

比如用户输入语句：lucene AND learned NOT hadoop。说明用户想找一个包含 lucene 和 learned 然而不包括 hadoop 的文档。

第二步：对查询语句进行分析。

由于查询语句有语法，因而要进行关键词分析和语法分析：

□ 关键词分析主要用来识别单词和关键字。如上述例子中，经过词法分析，得到的词有 lucene、learned、hadoop，关键字有 AND、NOT。如果在词法分析中发现不合法的关键词，则会出现错误。比如，lucene AMD learned，其中由于 AND 拼错，导致 AMD 作为一个普通的单词参与查询。

□ 语法分析主要是根据查询语句的语法规则来形成一棵语法树。如果发现查询语句不满足语法规则，则会报错。比如，lucene NOT AND learned，则会出错。如上述例子，lucene AND learned NOT hadoop 形成的语法树如图 5-44 所示。

第三步：搜索索引，得到符合语法树的文档。

□ 首先，在反向索引表中，分别找出包含 lucene、learned、hadoop 的文档链表。

□ 其次，对包含 lucene、learned 的链表进行合并操作，得到既包含 lucene 又包含 learned 的文档链表。

□ 最后，将此链表与 hadoop 的文档链表进行去

除操作，去除包含 hadoop 的文档，从而得到既包含 lucene 又包含 learned 而且不包含 hadoop 的文档链表。此文档链表就是我们要找的文档。

第四步：根据得到的文档和查询语句的相关性，对结果进行排序。

虽然在上一步，我们得到了想要的文档，然而对于查询结果应该按照与查询语句的相关性进行排序，相关率越高越靠前。如何计算文档和查询语句的相关性呢？不如我们把查询语句看做一片短小的文档，对文档与文档之间的相关性（relevance）进行打分（scoring），分数高的相关性好，就应该排在前面。那么如何对查询语句与文档的相关性进行打分呢？

首先一个文档有很多词（Term）组成，如 search、lucene、full-text、this、a、what 等。其次对于文档之间的关系，不同的 Term 重要性不同，例如对于本篇文档，search、lucene、full-text 就相对重要一些，this、a、what 可能相对不重要一些。所以如果两篇文档都包含 search，Lucene，fulltext，这两篇文档的相关性好一些，然而就算一篇文档包含 this、a、

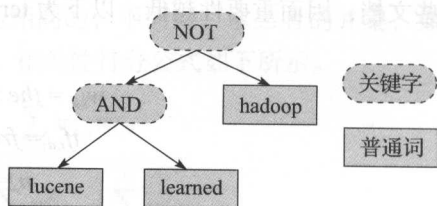


图 5-44 查询语句分析过程

what, 另一篇文档不包含 this、a、what, 也不能影响两篇文档的相关性。因而判断文档之间的关系, 首先找出哪些词 (Term) 对文档之间的关系最重要, 例如 search、scene、fulltext、然后判断这些词 (Term) 之间的关系。

找出词 (Term) 对文档的重要性的过程称为计算词的权重 (Term Weight) 的过程。计算词的权重 (term weight) 有两个参数, 第一个是词 (Term), 第二个是文档 (Document)。词的权重 (Term weight) 表示此词 (Term) 在此文档中的重要程度, 越重要的词 (Term) 有越大的权重 (Term weight), 因而在计算文档之间的相关性中将发挥更大的作用。判断词 (Term) 之间的关系从而得到文档相关性的过程应用一种叫做向量空间模型的算法 (Vector Space Model)。下面仔细分析一下这两个过程:

过程一: 计算权重 (Term weight) 的过程。影响一个词 (Term) 在一篇文档中的重要性主要有两个因素:

❑ Term Frequency (tf): 即此 Term 在此文档中出现了多少次。tf 越大说明越重要。

❑ Document Frequency (df): 即有多少文档包含此 Term。df 越大说明越不重要。

词 (Term) 在文档中出现的次数越多, 说明此词 (Term) 对该文档越重要, 例如“搜索”这个词, 在本文档中出现的次数很多, 说明本文档主要就是讲这方面的事的。然而在一篇英语文档中, this 出现的次数更多, 就说明越重要吗? 不是的, 这是由第二个因素进行调整, 第二个因素说明, 有越多的文档包含此词 (Term), 说明此词 (Term) 太普通, 不足以区分这些文档, 因而重要性越低。以下为 term weight 计算公式的简单典型实现。

$$w_{t,d} = tf_{t,d} \times \log(n/df_t)$$

$w_{t,d}$ = the weight of the term t in document d

$tf_{t,d}$ = frequency of term t in document d

n = total number of documents

df_t = the number of documents that contain term t

过程二: 判断 Term 之间的关系从而得到文档相关性的过程, 即向量空间模型的算法 (VSM)。

我们把文档看做一系列词 (Term), 每一个词 (Term) 都有一个权重 (Term Weight), 不同的词 (Term) 根据自己在文档中的权重来影响文档相关性的打分计算。于是我们把所有此文档中词 (Term) 的权重 (Term Weight) 看做一个向量。

Document = {term1, term2, ..., term N}

Document Vector = {weight1, weight2, ..., weight N}

同样, 我们把查询语句看做一个简单的文档, 也用向量来表示。

Query = {term1, term 2, ..., term N}

Query Vector = {weight1, weight2, ..., weight N}

把所有搜索出的文档向量及查询向量放到一个 N 维空间中, 每个词 (term) 是一维, 如图 5-45 所示。

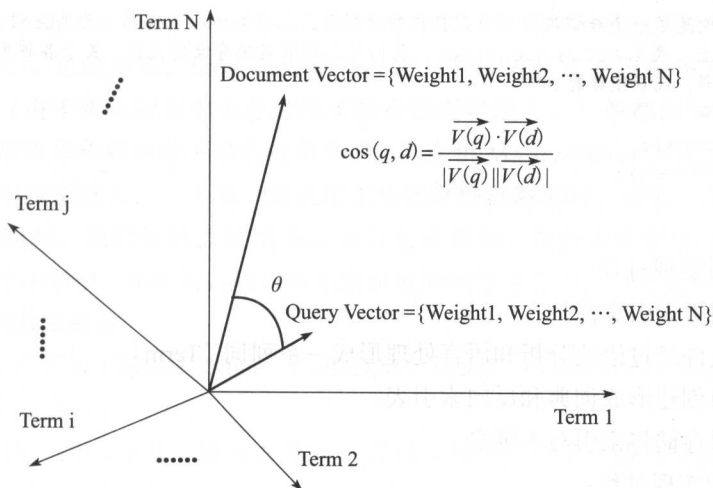


图 5-45 向量空间模型

我们认为两个向量之间的夹角越小，相关性越大。所以我们计算夹角的余弦值作为相关性的打分，夹角越小，余弦值越大，打分越高，相关性越大。但通常查询语句一般是很短的，包含的词（Term）是很少的，因而会出现查询向量的维数很小，而文档很长，包含词（Term）很多，文档向量维数很大的情况。

在这里，既然要放到相同的向量空间，自然维数是相同的，不同时，取二者的并集，如果不含某个词（Term）时，则权重（Term Weight）为 0。相关性打分公式如下所示。

$$\text{score}(q, d) = \frac{\vec{V}_q \cdot \vec{V}_d}{|\vec{V}_q| |\vec{V}_d|} = \frac{\sum_{i=1}^n w_{i,q} w_{i,d}}{\sqrt{\sum_{i=1}^n w_{i,q}^2} \sqrt{\sum_{i=1}^n w_{i,d}^2}}$$

虽然内部逻辑比较复杂，但经过 Lucene 的封装之后，具体代码实现如下：

```
# 打开存储位置
DirectoryReader ireader = DirectoryReader.open(directory);
# 创建搜索器
IndexSearcher isearcher = new IndexSearcher(ireader);
# 类似 SQL 进行关键字查询
QueryParser parser = new QueryParser(Version.LUCENE_CURRENT, "fieldname", analyzer);
Query query = parser.parse("text");
ScoreDoc[] hits = isearcher.search(query, null, 1000).scoreDocs;
assertEquals(1, hits.length);
for (int i = 0; i < hits.length; i++) {
    Document hitDoc = isearcher.doc(hits[i].doc);
    assertEquals("This is the text to be indexed.", hitDoc.get("fieldname"));
}
```

这里，我们创建了一个查询器，并设置其词法分析器，以及查询的“表名”为“fieldname”。查询结果会返回一个集合，类似 SQL 的 ResultSet，我们可以提取其中存储的内容。关于各种不同的查询方式，可以参考官方手册，或者推荐的 PPT

关闭查询器等

```
ireader.close();
```

```
directory.close();
```

总结

Lucene 索引实现过程：

- ❑ 有一系列被索引文件。
- ❑ 被索引文件经过语法分析和语言处理形成一系列词 (Term)。
- ❑ 经过索引创建形成词典和反向索引表。
- ❑ 通过索引存储将索引写入硬盘。

Lucene 搜索实现过程：

- ❑ 用户输入查询语句。
- ❑ 对查询语句进行语法分析和语言分析得到一系列词 (Term)。
- ❑ 通过语法分析得到一个查询树。
- ❑ 通过索引存储将索引读入到内存。
- ❑ 利用查询树搜索索引，从而得到每个词 (Term) 的文档链表，对文档链表进行交差，并得到结果文档。
- ❑ 利用搜索到的结果文档对查询的相关性进行排序。
- ❑ 返回查询结果给用户。
- ❑ 整个过程如图 5-46 所示。

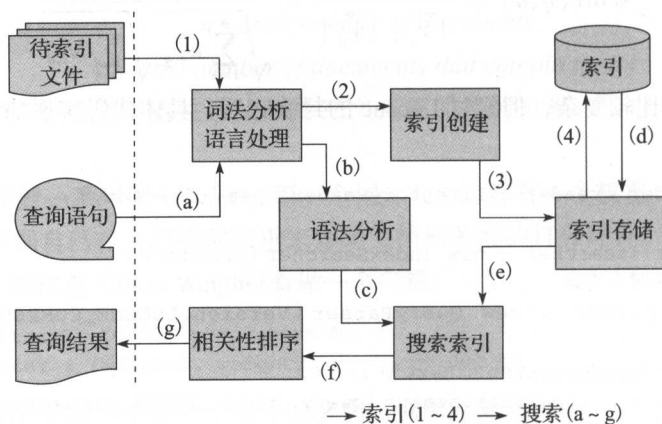


图 5-46 Lucene 索引实现流程图

2. Solr

Solr 是以 Lucene 为基础实现的文本检索应用服务，SolrCloud 是 Solr4.0 版本开发出的具

有开创意义的基于 Solr 和 Zookeeper 的分布式搜索方案，主要思想是使用 Zookeeper 作为集群的配置信息中心。也就是说，SolrCloud 是 Solr 的一种部署方式，除 SolrCloud 之外，Solr 还可以以单机版（由于实际应用中企业级全文检索数据量庞大，一般都以 SolrCloud 模式进行部署和应用，所以单机版在此不进行过多介绍）和多机 Master-Slaver 方式进行部署。分布式索引是指当索引越来越大，一个单一的系统无法满足磁盘需求时，或者一次简单的查询实在要耗费很多时间时，我们就可以使用 Solr 的分布式索引。在分布式索引中，原来的大索引，将会分成多个小索引，Solr 可以将这些小索引返回的结果合并，然后返回给客户端。

SolrCloud 的基本概念

SolrCloud 模式下有 Cluster、Node、Collection、Shard、LeaderCore、ReplicationCore 等重要概念。具体介绍如下：

- Cluster 集群：Cluster 是一组 Solr 节点，逻辑上作为一个单元进行管理，整个集群必须使用同一套 schema 和 SolrConfig。
- Node 节点：一个运行 Solr 的 JVM 实例。
- Collection：在 SolrCloud 集群中逻辑意义上的完整的索引，常常被划分为一个或多个 Shard，这些 Shard 使用相同的 Config Set，如果 Shard 数超过一个，那么索引方案就是分布式索引。SolrCloud 允许客户端用户通过 Collection 名称引用它，这样用户不需要关心分布式检索时需要使用的和 Shard 相关的参数。
- Core：也就是 SolrCore，一个 Solr 中包含一个或者多个 SolrCore，每个 SolrCore 可以独立提供索引和查询功能，SolrCore 的提出是为了增加管理灵活性和共用资源。SolrCloud 中使用的配置是在 Zookeeper 中，而传统的 SolrCore 的配置文件是在磁盘上的配置目录中。
- Config Set：SolrCore 提供服务必需的一组配置文件，每个 Config Set 有一个名字。最少需要包括 solrconfig.xml 和 schema.xml，除此之外，依据这两个文件的配置内容，可能还需要包含其他文件，例如中文索引需要的词库文件。Config Set 存储在 Zookeeper 中，可以重新上传或者使用 upconfig 命令进行更新，可使用 Solr 的启动参数 bootstrap_confdir 进行初始化或更新。
- Shard 分片：Collection 的逻辑分片。每个 Shard 被分成一个或者多个 Replicas，通过选举确定哪个是 Leader。
- Replica：Shard 的一个拷贝。每个 Replica 存在于 Solr 的一个 Core 中。换句话说，一个 SolrCore 对应着一个 Replica，例如一个命名为“test”的 collection 以 numShards=1 创建，并且指定 replicationFactor 为 2，这会产生 2 个 Replicas，也就是对应会有 2 个 Core，分别存储在不同的机器或者 Solr 实例上，其中一个会被命名为 test_shard1_replica1，另一个命名为 test_shard1_replica2，它们中的一个会被选举为 Leader。
- Leader：赢得选举的 Shard replicas，每个 Shard 有多个 Replicas，这几个 Replicas 需要选举来确定一个 Leader。选举可以发生在任何时间，但是通常它们仅在某个 Solr

实例发生故障时才会触发。当进行索引操作时，SolrCloud 会将索引操作请求传到此 Shard 对应的 Leader，Leader 再分发它们到全部 Shard 的 Replicas。

□ Zookeeper：Zookeeper 提供分布式锁功能，这对 SolrCloud 是必需的，主要负责处理 Leader 的选举。Solr 可以以内嵌的 Zookeeper 运行，也可以使用独立的 Zookeeper，并且 Solr 官方建议最好有 3 个以上的主机。

完整索引（Collection）逻辑，如图 5-47 所示。

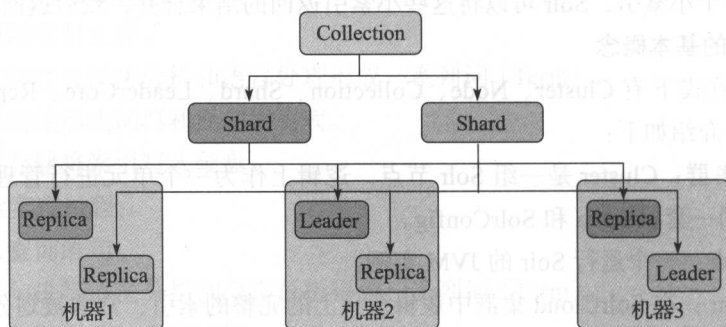


图 5-47 完整索引（Collection）逻辑图

在 SolrCloud 模式下 Collection 是访问 Cluster 的入口，这个入口有什么用呢？比如，集群中有好多台机器，那么访问这个集群通过哪个地址呢，必须有一个接口地址，Collection 就是这个接口地址。可见 Collection 是一个逻辑存在的东西，因此是可以跨 Node 的，在任意节点上都可以访问 Collection。Shard 其实也是逻辑存在的，因此 Shard 也是可以跨 Node 的；1 个 Shard 下面可以包含 0 个或者多个 Replication，但 1 个 Shard 下面能且只能包含一个 Leader，如果 Shard 下面的 Leader 挂掉了，会从 Replication 中再选举一个 Leader。

此处需要注意的是在 Solr4.0 中，可以在 Solr AdminGUI 中增加和删除 Core，如果 Shard 中最后一个 Core 被删除了，Shard 是不会自动删除的，这会导致集群出错，而且如果 Shard 中所有的 Core 宕机了，会导致不能继续插入新的记录，从而导致查询会受到影响，其实如果一个 Shard 下的所有 Core 宕机了，SolrCloud 应该允许插入到其他存活的 Shard 中，这在后期版本中的 Solr 中应该会被支持。

索引操作基本架构

如图 5-48 所示为拥有 4 个 Solr 节点的集群，索引分布在两个 Shard 中，每个 Shard 包含两个 Solr 节点，一个是 Leader 节点，一个是 Replica 节点，此外集群中有一个负责维护集群状态信息的 Overseer 节点，它是一个总控制器。

集群的所有状态信息都放在 Zookeeper 集群中统一维护。从图 5-49 中还可以看到，任何一个节点都可以接收索引创建或者更新的请求，然后再将这个请求转发到索引文档所应该属于的那个 Shard 的 Leader 节点，Leader 节点更新结束完成后，最后将版本号和文档转发给同属于一个 Shard 的 Replicas 节点。

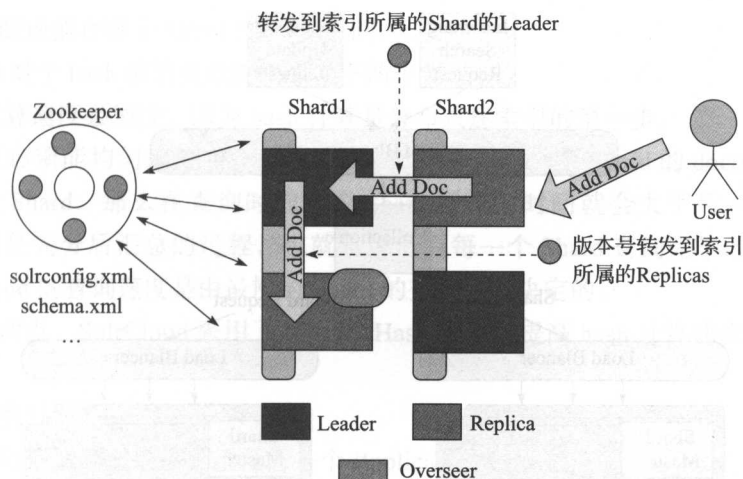


图 5-48 索引操作基本架构

工作模式

首先来看下索引和 Solr 实体对照图 5-49。

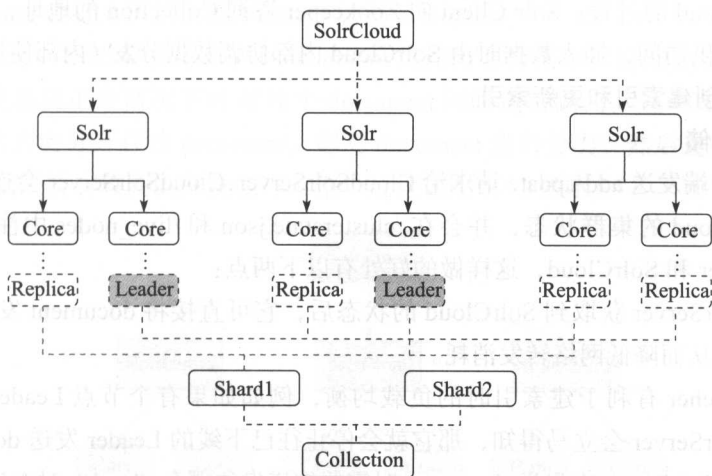


图 5-49 索引和 Solr 实体对照图

SolrCloud 中包含多个 Solr Instance，而每个 Solr Instance 中包含多个 SolrCore，SolrCore 对应着一个可访问的 Solr 索引资源，每个 SolrCore 对应着一个 Replica 或者 Leader，这样，当 Solr Client 通过 Collection 访问 Solr 集群时，便可通过 Shard 分片找到对应的 Replica 即 SolrCore，从而就可以访问索引文档了。

在 SolrCloud 模式下，同一个集群中所有 Core 的配置是统一的，Core 有 Leader 和 Replication 两种角色，每个 Core 一定属于一个 Shard，Core 在 Shard 中扮演 Leader 还是 Replication 由 Solr 内部的 Zookeeper 自动协调（如图 5-50 所示）。

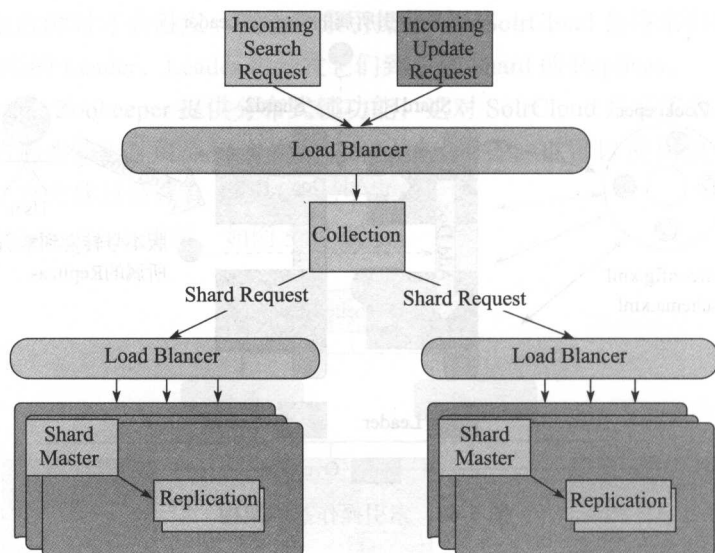


图 5-50 SolrCloud 工作模式

访问 SolrCloud 的过程：Solr Client 向 Zookeeper 咨询 Collection 的地址，Zookeeper 返回存活的节点地址供访问，插入数据时由 SolrCloud 内部协调数据分发（内部使用一致性哈希）。

SolrCloud 创建索引和更新索引

（1）索引存储

当 Solr 客户端发送 add/update 请求给 CloudSolrServer，CloudSolrServer 会连接至 Zookeeper 获取当前 SolrCloud 的集群状态，并会在 /clusterstate.json 和 /live_nodes 中注册 watcher，便于监视 Zookeeper 和 SolrCloud，这样做的好处有以下两点：

- ❑ CloudSolrServer 获取到 SolrCloud 的状态后，它可直接将 document 发往 SolrCloud 的 Leader，从而降低网络转发消耗。
- ❑ 注册 watcher 有利于建索引时的负载均衡，例如如果有个节点 Leader 下线了，那么 CloudSolrServer 会立马得知，那它就会停止往已下线的 Leader 发送 document。

此外，CloudSolrServer 在发送 document 时需要知道发往哪个 Shard？对于建好的 SolrCloud 集群，每一个 Shard 都会有一个 hash 区间，当 document 进行 update 时，SolrCloud 就会计算这个 document 的 hash 值，然后根据该值和 Shard 的 hash 区间来判断这个 document 应该发往哪个 Shard，Solr 使用 documentroute 组件来进行 document 的分发。目前，Solr 有两个 DocRouter 类的子类：CompositeIdRouter（Solr 默认采用的）类和 ImplicitDocRouter 类，当然我们也可以通过继承 DocRouter 来定制化我们的 document route 组件。

举例来说当 Solr Shard 建立时，Solr 会给每一个 Shard 分配 32bit 的 hash 值区间，例如 SolrCloud 有两个 Shard 分别为 A、B，那么 A 的 hash 值区间就为 80000000-ffffff，B 的 Hash 值区间为 0-7ffffff。默认的 CompositeIdRouter hash 策略会根据 document ID 计算出唯一的

hash 值, 并判断该值在哪个 Shard 的 hash 区间内。

SolrCloud 对于 hash 值的获取提出了以下两个要求:

- hash 计算速度必须快, 因为 hash 计算是分布式建索引的第一步。
- hash 值必须能均匀地分布于每一个 Shard, 如果有一个 Shard 的 document 数量大于另一个 Shard, 那么在查询时前一个 Shard 所花的时间就会大于后一个, SolrCloud 的查询是先后汇总的过程, 也就是说最后每一个 Shard 查询完毕才算完毕, 所以 SolrCloud 的查询速度是由最慢的 Shard 的查询速度决定的。

基于以上两点, SolrCloud 采用了 MurmurHash 算法以提高 hash 计算速度和 hash 值的均匀分布。

(2) 创建索引

- 用户可以把新建文档提交给任意一个 Replica (SolrCore)。
- 如果它不是 Leader, 则会把请求转给和自己同 Shard 的 Leader。
- Leader 把文档路由给本 Shard 的每个 Replica。
- 如果文档基于路由规则 (例如取 hash 值) 并不属于当前的 Shard, Leader 会把它转交给对应 Shard 的 Leader。
- 对应 Leader 会把文档路由给本 Shard 的每个 Replica。

需要注意的是, 添加索引时, 单个 document 的路由非常简单, 但是 SolrCloud 支持批量添加索引, 也就是说正常情况下可对 N 个 document 同时进行路由。这时 SolrCloud 会根据 document 路由的去向分开存放 document, 即对 document 进行分类, 然后进行并发发送至相应的 Shard, 这就需要有较高的并发能力。整个过程如图 5-51 所示。

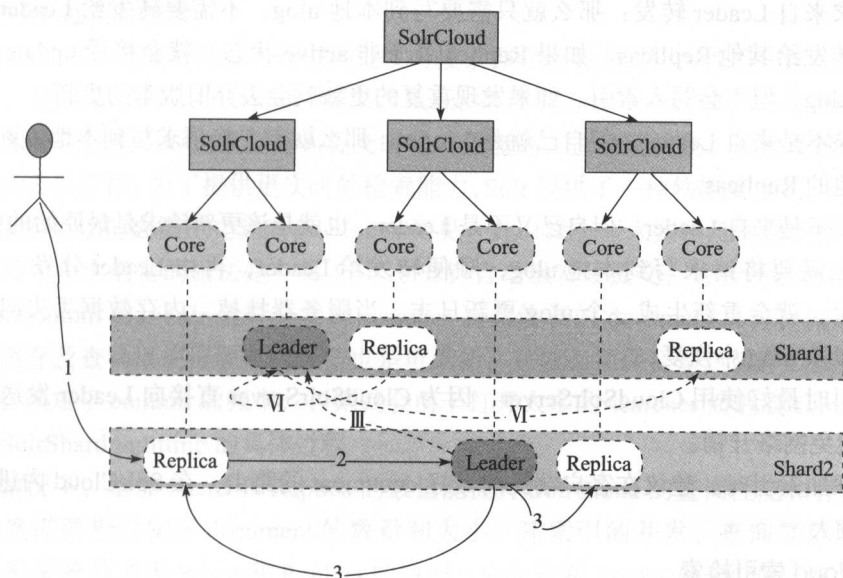


图 5-51 SolrCloud 添加索引工作模式

（3）更新索引

Leader 接受到 update 请求后，先将 update 信息存放到本地的 update log，同时 Leader 还会给 document 分配新的 version，对于已存在的 document，如果新的版本高就会抛弃旧版本，最后发送至 Replica。

一旦 document 经过验证以及加入 version 后，就会并行地被转发至所有上线的 replica。SolrCloud 并不会关注那些已经下线的 Replica，因为当它们上线时会有 recovery 进程对其进行恢复。如果转发的 Replica 处于 recovering 状态，那么这个 Replica 就会把 update 放入 update-transaction 日志。

当 Leader 接受到所有 Replica 的反馈成功后，它才会反馈客户端成功。只要 Shard 中有一个 Replica 是 active 的，Solr 就会继续接受 update 请求。这一策略其实是牺牲了一致性换取了写入的有效性。这其中有一个重要参数：leaderVoteWait 参数，它表示当只有一个 Replica 时，这个 Replica 会进入 recovering 状态并持续一段时间等待 Leader 的重新上线。如果在这段时间内 Leader 没有上线，那么它就会转成 Leader，其中可能会有一些 document 丢失。当然可以使用 majority quorum 来避免这个情况，这跟 Zookeeper 的 Leader 选举策略一样，例如当多数的 Replica 下线了，那么客户端的 write 就会失败。

索引的 commit 有两种，一种是 softcommit，即在内存中生成 segment，document 是可见的（可查询到）但是没写入磁盘，断电后数据会丢失。另一种是 hardcommit，直接将数据写入磁盘且数据可见。

对 Solr 更新索引和创建索引的几点总结：

Leader 转发的规则如下：

- ❑ 请求来自 Leader 转发：那么就只需要写到本地 ulog，不需要转发给 Leader，也不需要转发给其他 Replicas。如果 Replica 处于非 active 状态，就会接受 update 请求并写入 ulog，但不会写入索引。如果发现重复的更新就会丢弃旧版本的更新。
- ❑ 请求不是来自 Leader，但自己就是 Leader，那么就需要将请求写到本地，顺便分发给其他的 Replicas。
- ❑ 请求不是来自 Leader，但自己又不是 Leader，也就是该更新请求是最原始的更新请求，那么需要将请求写到本地 ulog，顺便转发给 Leader，再由 Leader 分发。每 commit 一次，就会重新生成一个 ulog 更新日志，当服务器挂掉，内存数据丢失时，数据就可以从 ulog 中恢复。

建索引时最好使用 CloudSolrServer，因为 CloudSolrServer 直接向 Leader 发送 update 请求，从而避免网络开销。

批量添加索引时，建议在客户端提前做好 document 的路由，在 SolrCloud 内进行文档路由，开销较大。

SolrCloud 索引检索

在创建好索引的基础上，SolrCloud 检索索引相对就比较简单了，如图 5-52 所示。

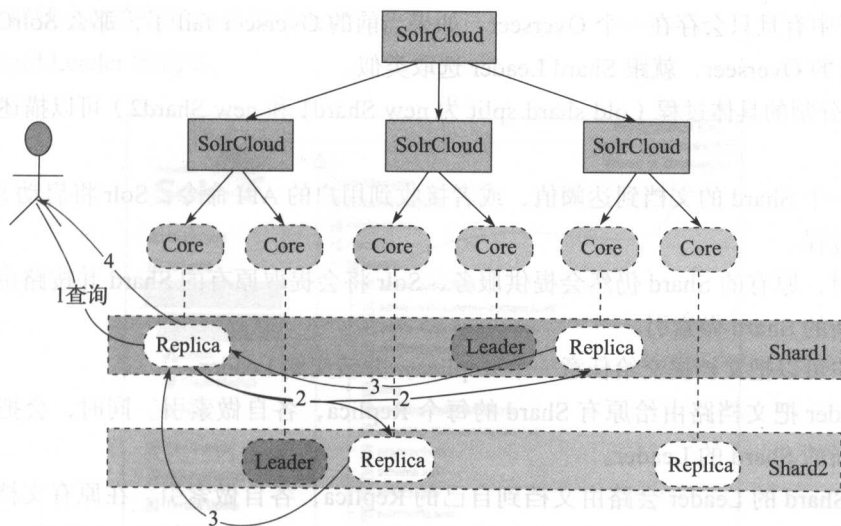


图 5-52 SolrCloud 索引检索

- 用户的一个查询，可以发送到含有该 Collection 的任意 Solr 的 Server，Solr 内部处理的逻辑会转到一个 Replica。
- 此 Replica 会基于查询索引的方式，启动分布式查询，基于索引的 Shard 的个数，把查询转为多个子查询，并把每个子查询定位到对应 Shard 的任意一个 Replica。
- 每个子查询返回查询结果。
- 最初的 Replica 合并子查询，并把最终结果返回给用户。

(1) NRT 近实时搜索

SolrCloud 支持近实时搜索，所谓的近实时搜索即在较短的时间内使得新添加的 document 可见可查，这主要基于 softcommit 机制（Lucene 是没有 softcommit 的，只有 hardcommit）。上面提到 Solr 创建索引时的数据是在提交时写入磁盘的，这是硬提交，硬提交确保了即便是停电也不会丢失数据；为了提供更实时的检索能力，Solr 提供了一种软提交方式。软提交（soft commit）指的是仅把数据提交到内存，index 可见，此时没有写入到磁盘索引文件中。

在设计中一个通常的做法是：每 1 ~ 10 分钟自动触发硬提交，每秒钟自动触发软提交，当进行 softcommit 时，Solr 会打开新的 searcher 从而使新的 document 可见，同时 Solr 还会进行预热缓存及查询以使得缓存的数据也是可见的，这就必须保证预热缓存以及预热查询的执行时间必须短于 commit 的频率，否则就会由于打开太多的 searcher 而造成 commit 失败。

(2) SolrShard Splitting 的具体过程

一般情况下，增加 Shard 和 Replica 的数量能提升 SolrCloud 的查询性能和容灾能力，但是我们仍然需要根据实际 document 的数量和大小、建索引的并发、查询复杂度，以及索引的增长率来统筹考虑 Shard 和 Replica 的数量。Solr 依赖 Zookeeper 实现集群的管理，在 Zookeeper 中有一个 Znode 是 /clusterstate.json，它存储了当前时刻下整个集群的状态。同时

在一个集群中有且只会存在一个 Overseer，如果当前的 Overseer fail 了，那么 SolrCloud 就会选出一个新的 Overseer，就跟 Shard Leader 选取类似。

Shard 分割的具体过程（old shard split 为 new Shard1 和 new Shard2）可以描述为图 5-53 的图例：

- ❑ 当一个 Shard 的文档到达阈值，或者接收到用户的 API 命令，Solr 将启动 Shard 的分裂过程。
- ❑ 此时，原有的 Shard 仍然会提供服务，Solr 将会提取原有的 Shard 并按路由规则，转到新的 Shard 做索引。
- ❑ 用户可以把文档提交给任意一个 Replica，并转交给 Leader。
- ❑ Leader 把文档路由给原有 Shard 的每个 Replica，各自做索引。同时，会把文档路由给新的 Shard 的 Leader。
- ❑ 新 Shard 的 Leader 会路由文档到自己的 Replica，各自做索引，在原有文档重新索引完成，系统会把分发文档路由切换到对应新的 Leader 上，原有 Shard 关闭。Shard 只是一个逻辑概念，所以 Shard 的 Splitting 只是将原有 Shard 的 Replica 均匀地分不到更多 Shard 的更多 Solr 节点上。

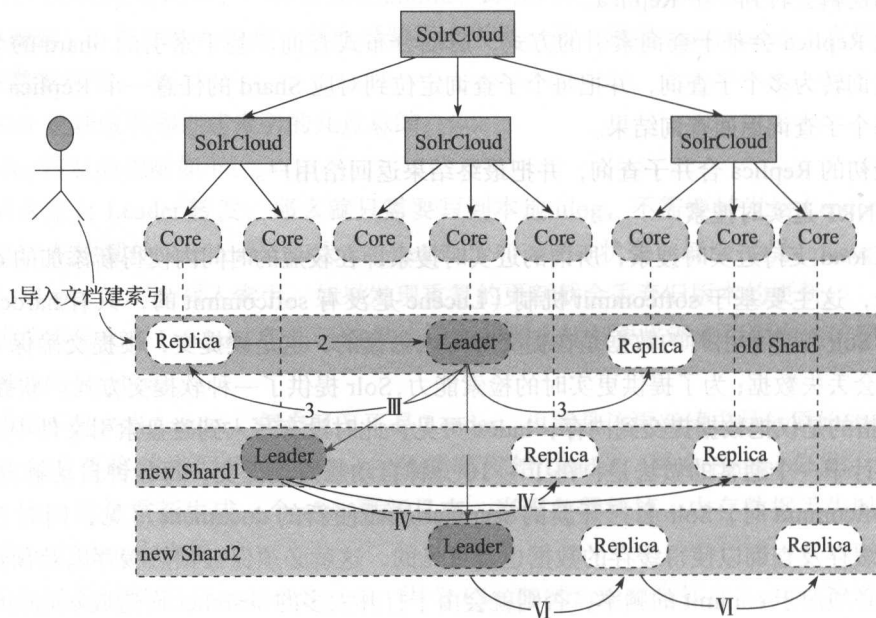


图 5-53 SolrShard 分割过程

SolrCloud 配置 Zookeeper 集群

SolrCloud 中使用 ZooKeeper 主要实现以下三点功能：

- ❑ 集中配置存储以及管理，如图 5-54 所示。

□ 集群状态改变时进行监控以及通知。

□ Shard Leader 的选举。

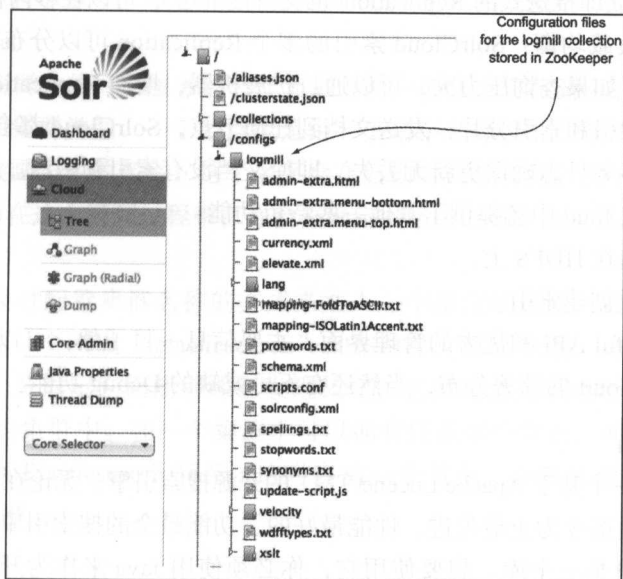


图 5-54 SolrCloud 集群配置

示例：包含两个 Shard 的集群，带 Shard 备份和 Zookeeper 集群机制，如图 5-55 所示。

对 Zookeeper 服务器也设置一个集群，让其具备高可用性和容错性。有两种方式可选，一种是提供一个外部独立的 Zookeeper 集群，另一种是每个 Solr 服务器都启动一个内嵌的 Zookeeper 服务器，再将这些 Zookeeper 服务器组成一个集群。

总结

通过以上介绍，可看出 SolrCloud 保证了整个 Solr 应用的 High Availability。

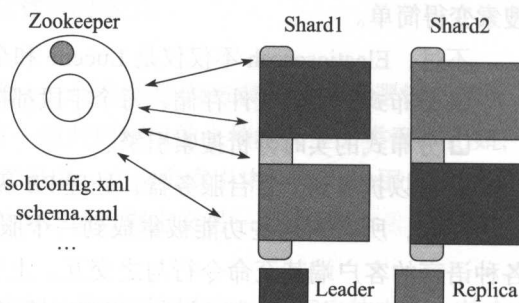


图 5-55 SolrCloud 集群机制

□ 集中式的配置信息使用 Zookeeper 进行集中配置。启动时可以指定把 Solr 的相关配置文件上传 Zookeeper，多机器共用。这些 Zookeeper 中的配置不会再拿到本地缓存，Solr 直接读取 Zookeeper 中的配置信息。另外配置文件的变动，所有机器都可以感知到，Solr 的一些任务也是通过 Zookeeper 作为媒介发布的，目的是为了容错，这使得 Solr 接收到任务，但在执行任务时崩溃的机器，在重启后，或者集群选出候选者时，可以再次执行这个未完成的任务。

□ SolrCloud 对索引分片，并对每个分片创建多个 Replication。每个 Replication 都可以

对外提供服务。一个 Replication 挂掉不会影响索引服务，更强大的是，SolrCloud 还能自动地在其他机器上帮你把失败机器上的索引 Replication 重建并投入使用。

- 近实时搜索：立即推送式的 Replication(也支持慢推送)，可以在秒内检索到新加入索引。
- 查询时自动负载均衡：SolrCloud 索引的多个 Replication 可以分布在多台机器上，均衡查询压力，如果查询压力大，可以通过扩展机器，增加 Replication 来减缓。
- 自动分发的索引和索引分片：发送文档到任何节点，SolrCloud 都会转发到正确节点。
- 事务日志：事务日志确保更新无丢失，即使文档没有索引到磁盘。

除此之外，SolrCloud 中还提供了其他一些特色功能：

- 可将索引存储在 HDFS 上。
- 通过 MR 批量创建索引。
- 强大的 RESTful API 和优秀的管理界面，主要信息一目了然，可以清晰地以图形化方式看到 SolrCloud 的部署分布，当然还有不可或缺的 Debug 功能。

3. Elasticsearch

Elasticsearch 是一个基于 Apache Lucene(TM) 的开源搜索引擎。无论在开源还是专有领域，Lucene 可以被认为是迄今为止最先进、性能最好的、功能最全的搜索引擎库。

但是，Lucene 只是一个库。想要使用它，你必须使用 Java 来作为开发语言并将其直接集成到你的应用中。更糟糕的是，Lucene 非常复杂，你需要深入了解检索的相关知识来理解它是如何工作的。Elasticsearch 也使用 Java 开发并使用 Lucene 作为其核心来实现所有索引和搜索的功能，但是它的目的是通过简单的 RESTful API 来隐藏 Lucene 的复杂性，从而让全文搜索变得简单。

不过，Elasticsearch 不仅仅是 Lucene 和全文搜索，我们还能这样去描述它：

- 分布式的实时文件存储，每个字段都被索引并可被搜索、
- 分布式的实时分析搜索引擎、
- 可以扩展到上百台服务器，处理 PB 级结构化或非结构化数据。

而且，所有的这些功能被集成到一个服务中，你的应用可以通过简单的 RESTful API、各种语言的客户端甚至命令行与之交互。上手 Elasticsearch 非常容易，它提供了许多合理的缺省值，并对初学者隐藏了复杂的搜索引擎理论。它开箱即用（安装即可使用），只需很少的学习即可在生产环境中使用。

Elasticsearch 的基本介绍

Elasticsearch 有几个核心概念。从一开始理解这些概念会对整个学习过程有莫大的帮助。

(1) 接近实时 (NRT)

Elasticsearch 是一个接近实时的搜索平台。这意味着，从索引一个文档直到这个文档能够被搜索到有一个轻微的延迟（通常是 1 秒）。

(2) 集群 (Cluster)

一个集群就是由一个或多个节点组织在一起，它们共同持有你整个的数据，并一起提供

索引和搜索功能。一个集群由一个唯一的名字标识,这个名字默认就是“elasticsearch”。这个名字是重要的,因为一个节点只能通过指定某个集群的名字,来加入这个集群。在产品环境中显式地设定这个名字是一个好习惯,但是使用默认值来进行测试/开发也是不错的。

(3) 节点 (Node)

一个节点是你集群中的一个服务器,作为集群的一部分,它存储你的数据,参与集群的索引和搜索功能。和集群类似,一个节点也是由一个名字来标识的,默认情况下,这个名字是一个随机的漫威漫画角色的名字,这个名字会在启动时赋予节点。这个名字对于管理工作来说挺重要的,因为在这个管理过程中,你会去确定网络中的哪些服务器对应于 Elasticsearch 集群中的哪些节点。

一个节点可以通过配置集群名称的方式来加入一个指定的集群。默认情况下,每个节点都会被安排加入到一个叫做“elasticsearch”的集群中,这意味着,如果你在你的网络中启动了若干个节点,并假定它们能够相互发现彼此,它们将会自动地形成并加入到一个叫做“elasticsearch”的集群中。在一个集群中可以拥有任意多个节点。而且,如果当前你的网络中没有运行任何 Elasticsearch 节点,这时启动一个节点,会默认创建并加入一个叫做“elasticsearch”的集群。

(4) 索引 (Index)

一个索引就是一个拥有几分相似特征的文档的集合。比如,你可以有一个客户数据的索引,另一个产品目录的索引,还有一个订单数据的索引。一个索引由一个名字来标识(必须全部是小写字母的),并且当我们要对对应于这个索引中的文档进行索引、搜索、更新和删除时,都要使用到这个名字,一个集群中可以定义任意多的索引。

(5) 类型 (Type)

在一个索引中,你可以定义一种或多种类型。一个类型是你的索引的一个逻辑上的分类/分区,其语义完全由你来定。通常,会为具有一组共同字段的文档定义一个类型。比如,假设你运营一个博客,将所有数据存储到一个索引中。在这个索引中,可以为用户数据定义一个类型,为文章数据定义另一个类型,也可以为评论数据定义另一个类型。

(6) 文档 (Document)

一个文档是一个可被索引的基础信息单元。比如,你可以拥有某一个客户的文档,某个产品的一个文档,当然,也可以拥有某个订单的一个文档。文档以 JSON (Javascript Object Notation) 格式来表示,而 JSON 是一个到处存在的互联网数据交互格式。在一个 Index/Type 中可以存储任意多的文档。注意,尽管一个文档,物理上存在于一个索引之中,文档必须被索引/赋予一个索引的 Type。

(7) 分片 (Shards)

分片 (Shard) 是一个最小级别的“工作单元”(Worker Unit),它保存了索引中所有数据的一部分。一个索引可以存储超出单个节点硬件限制的大量数据。比如,一个具有 10 亿文档的索引占据 1TB 的磁盘空间,而任一节点都没有这样大的磁盘空间;或者单个节点处理搜

索请求，响应太慢。

为了解决这个问题，Elasticsearch 提供了将索引划分成多份的能力，这些份就叫做分片。当你创建一个索引时，你可以指定你想要的分片数量。每个分片本身也是一个功能完善并且独立的“索引”，这个“索引”可以被放置到集群中的任何节点上。分片可以是主分片（Primary Shard）或者是复制分片（Replica Shard）。你索引中的每个文档属于一个单独的主分片，所以主分片的数量决定了索引最多能存储多少数据。

分片有两个重要特点：①允许你水平分割 / 扩展你的内容容量；②允许你在分片（潜在地，位于多个节点上）之上进行分布式的、并行的操作，进而提高性能 / 吞吐量。至于一个分片怎样分布，它的文档怎样聚合成搜索请求，是完全由 Elasticsearch 管理的，对于作为用户的你来说，这些都是透明的，并且每个索引都可以被分成多个分片。

（8）复制（Replicas）

在一个网络 / 云的环境中，失败随时都可能发生，在某个分片 / 节点不知为什么就处于离线状态，或者由于任何原因消失了，这种情况下，有一个故障转移机制是非常有用并且是强烈推荐的。为此目的，Elasticsearch 允许你创建分片的一份或多份拷贝，这些拷贝叫做复制分片，或者直接叫复制。复制分片只是主分片的一个副本，它可以防止硬件故障导致的数据丢失，同时可以提供读请求，例如搜索或者从别的 Shard 取回文档。

关于复制的两个重要特点：①在分片 / 节点失败的情况下，提供了高可用性。因为这个原因，注意到复制分片从不与原 / 主要（Original/Primary）分片置于同一节点上是非常重要的。②扩展你的搜索量 / 吞吐量，因为搜索可以在所有的复制上并行运行。

一个索引也可以被复制 0 次（意思是没有复制）或多次。一旦复制了，每个索引就有了主分片（作为复制源的原来的分片）和复制分片（主分片的拷贝）之别。分片和复制的数量可以在索引创建时指定。在索引创建之后，你可以在任何时候动态地改变复制的数量，但是你事后不能改变分片的数量。

默认情况下，Elasticsearch 中的每个索引被分配 5 个主分片和 1 个复制分片，这意味着，如果你的集群中有至少两个节点，你的索引将会有 5 个主分片和另外 5 个复制分片（1 个完全拷贝），这样的话每个索引总共就有 10 个分片。

分布式集群

Elasticsearch 用于构建高可用和可扩展的系统。扩展的方式可以是购买更好的服务器（纵向扩展，vertical scale or scaling up）或者购买更多的服务器（横向扩展，horizontal scale or scaling out）。Elasticsearch 虽然能从更强大的硬件中获得更好的性能，但是纵向扩展有它的局限性。真正的扩展应该是横向的，它通过增加节点来均摊负载和增加可靠性。

对于大多数数据库而言，横向扩展意味着你的程序将做非常大的改动才能利用这些新添加的设备。对比来说，Elasticsearch 天生就是分布式的：它知道如何管理节点来提供高扩展和高可用。这意味着你的程序不需要关心这些。在这里我们将介绍如何创建集群（Cluster）、节点（Node）和分片（Shards），使其按照你的需求进行扩展，并保证在硬件故障时数据依旧安全。

(1) 空集群

如果我们启动一个单独的节点，它还没有数据和索引，这个集群看起来就如图 5-56 所示。

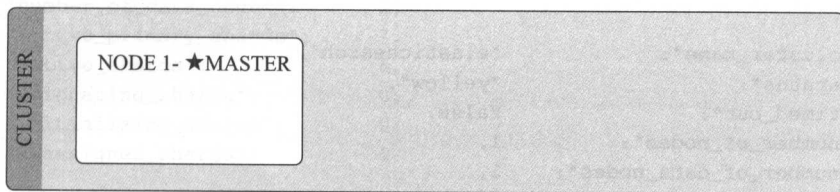


图 5-56 只有一个空节点的集群

一个节点 (Node) 就是一个 Elasticsearch 实例，而一个集群 (Cluster) 由一个或多个节点组成，它们具有相同的 `cluster.name`，它们协同工作，分享数据和负载。当加入新的节点或者删除一个节点时，集群就会感知到并平衡数据。

集群中一个节点会被选举为主节点 (Master)，它将临时管理集群级别的一些变更，例如新建或删除索引、增加或移除节点等。主节点不参与文档级别的变更或搜索，这意味着在流量增长时，该主节点不会成为集群的瓶颈。任何节点都可以成为主节点。我们例子中的集群只有一个节点，所以它会充当主节点的角色。

作为用户，我们能够与集群中的任何节点通信，包括主节点。每一个节点都知道文档存在于哪个节点上，它们可以转发请求到相应的节点上。我们访问的节点负责收集各节点返回的数据，最后一起返回给客户端。这一切都由 Elasticsearch 处理。

(2) 添加索引

为了将数据添加到 Elasticsearch，我们需要索引 (Index)——一个存储关联数据的地方。实际上，索引只是一个用来指向一个或多个分片 (Shards) 的“逻辑命名空间”(Logical Namespace)。

让我们在集群中唯一一个空节点上创建一个叫做 `blogs` 的索引。默认情况下，一个索引被分配 5 个主分片，但是为了演示的目的，我们只分配 3 个主分片和 1 个复制分片 (每个主分片都有一个复制分片)，如图 5-57 所示。

```
PUT /blogs
{
  "settings" : {
    "number_of_shards" : 3,
    "number_of_replicas" : 1
  }
}
```

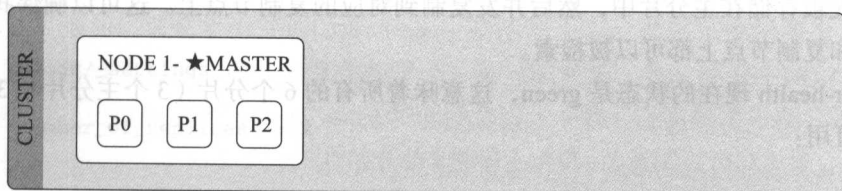


图 5-57 附带索引的单一节点集群

我们的集群现在看起来就像图 5-57——3 个主分片都被分配到 Node 1。如果我们现在检查集群健康（cluster-health），我们将看到以下信息：

```
{
  "cluster_name":      "elasticsearch",
  "status":            "yellow",
  "timed_out":         false,
  "number_of_nodes":   1,
  "number_of_data_nodes": 1,
  "active_primary_shards": 3,
  "active_shards":     3,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 3
}
```

集群的状态现在是 yellow，我们的 3 个复制分片还没有被分配到节点上。

集群的健康状态 yellow 表示所有的主分片（Primary Shards）启动并且正常运行了——集群已经可以正常处理任何请求——但是复制分片（Replica Shards）还没有全部可用。事实上，所有的 3 个复制分片现在都是 unassigned 状态——它们还未被分配给节点。在同一个节点上保存相同的数据副本是没有必要的，如果这个节点出现故障，那所有的数据副本也会丢失。

（3）故障转移

在单一节点上运行意味着有单点故障的风险——没有数据备份。幸运的是，要防止单点故障，我们唯一需要做的就是启动另一个节点。如果我们启动了第二个节点，这个集群看起来就如图 5-58 所示：双节点集群——所有的主分片和复制分片都已分配。

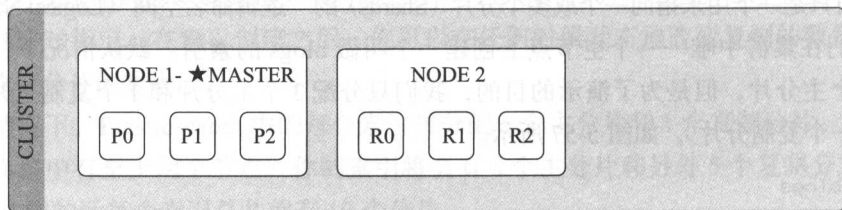


图 5-58 双节点集群

第二个节点已经加入集群，3 个复制分片（Replica Shards）也已经被分配了——分别对应 3 个主分片，这意味着在丢失任意一个节点的情况下依旧可以保证数据的完整性。文档的索引将首先被存储在主分片中，然后并发复制到对应的复制节点上。这可以确保我们的数据在主节点和复制节点上都可以被检索。

cluster-health 现在的状态是 green，这意味着所有的 6 个分片（3 个主分片和 3 个复制分片）都已可用：

```
{
  "cluster_name":      "elasticsearch",
```

```

"status": "green",
"timed_out": false,
"number_of_nodes": 2,
"number_of_data_nodes": 2,
"active_primary_shards": 3,
"active_shards": 6,
"relocating_shards": 0,
"initializing_shards": 0,
"unassigned_shards": 0
}

```

集群的状态是 **green**。此时，我们的集群不仅是功能完备的，而且是高可用的。

(4) 集群扩展

随着应用需求的增长，我们该如何扩展，如果我们启动第三个节点，我们的集群会重新组织自己，图 5-59 中包含 3 个节点的集群——分片已经被重新分配以平衡负载。

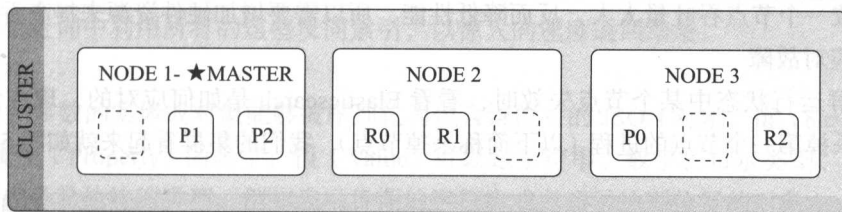


图 5-59 具有 3 个节点的集群

Node 3 包含了分别来自 Node 1 和 Node 2 的一个分片，这样每个节点就有两个分片，和之前相比少了一个，这意味着每个节点上的分片将获得更多的硬件资源（CPU、RAM、I/O）。

分片本身就是一个完整的搜索引擎，它可以使用单一节点的所有资源。我们拥有 6 个分片（3 个主分片和 3 个复制分片），最多可以扩展到 6 个节点，每个节点上有一个分片，每个分片可以 100% 使用这个节点的资源。

主分片的数量在创建索引时已经确定。实际上，这个数量定义了能存储到索引中数据的最大数量（实际的数量取决于你的数据、硬件和应用场景）。然而，主分片或者复制分片都可以处理读请求——搜索或文档检索，所以数据的冗余越多，我们能处理的搜索吞吐量就越大。

继续把集群扩展到 6 个以上的节点，复制分片的数量可以在运行中的集群中动态地变更，这允许我们可以根据需求扩大或者缩小规模。让我们把复制分片的数量从原来的 1 增加到 2：

```

PUT /blogs/_settings
{
  "number_of_replicas" : 2
}

```

增加 `number_of_replicas` 到 2 时如图 5-60 所示。

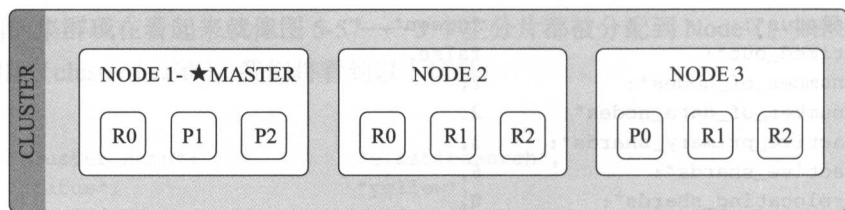


图 5-60 增加复制分片数据

从图 5-60 中可以看出，blogs 索引现在有 9 个分片：3 个主分片和 6 个复制分片。这意味着我们能够扩展到 9 个节点，再次变成每个节点一个分片。这样使我们的搜索性能相比原始的三节点集群增加三倍。

需要注意的是，在同样数量的节点上增加更多的复制分片并不能提高性能，因为这样做平均每个分片所占有的硬件资源就减少了。这种情况下，大部分请求都聚集到了分片较少的节点，导致一个节点吞吐量太大，反而降低性能，所以需要增加硬件资源来提高吞吐量。

（5）应对故障

当集群运行状态中某个节点失效时，看看 Elasticsearch 是如何应对的，现在尝试一下。如果我们杀掉第一个节点的进程（以下简称杀掉节点），我们的集群看起来就如图 5-61 所示。

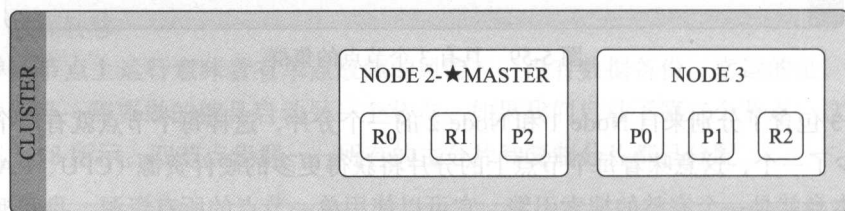


图 5-61 故障时集群状态

我们杀掉的节点是一个主节点。一个集群必须要有一个主节点才能使其功能正常，所以集群做的第一件事就是各节点选举了一个新的主节点：Node 2。主分片 1 和 2 在我们杀掉 Node 1 时已经丢失，我们的索引在丢失主分片时不能正常工作。如果此时我们检查集群健康，我们将看到状态 red：不是所有主节点都可用！

幸运的是，丢失的两个主分片的完整拷贝存在于其他节点上，所以新主节点做的第一件事是把这些在 Node 2 和 Node 3 上的复制分片升级为主分片，这时集群健康回到 yellow 状态。这个提升是瞬间完成的，就好像按了一下开关。

为什么集群健康状态是 yellow 而不是 green？我们三个主分片，但是我们指定了每个主分片对应两个复制分片，当前却只有一个复制分片被分配，这就是集群状态无法达到 green 的原因，不过不用太担心这个：当我们杀掉 Node 2，我们的程序依然可以在没有丢失数据的情况下继续运行，因为 Node 3 还有每个分片的拷贝。

如果我们重启 Node 1，集群将能够重新分配丢失的复制分片，集群状况与上面“增加

number_of_replicas 到 2”类似。如果 Node 1 依旧有旧分片的拷贝，它将会尝试再利用它们，它只会从主分片上复制在故障期间有数据变更的那一部分。

数据

无论程序怎么写，意图是一样的：组织数据为我们的目标所服务。但数据并不只是由随机比特和字节组成，我们在数据节点间建立关联来表示现实世界中的实体或者“某些东西”。Elasticsearch 作为一个分布式的文档（Document）存储引擎。它可以实时存储并检索复杂数据结构——序列化的 JSON 文档。一旦文档被存储在 Elasticsearch 中，它就可以在集群的任一节点上被检索。

当然，我们不仅需要存储数据，还要快速地批量查询。虽然已经有很多 NoSQL 的解决方案允许我们以文档的形式存储对象，但它们依旧需要考虑如何查询这些数据，以及哪些字段需要被索引以便检索时更加快速。在 Elasticsearch 中，每一个字段的数据都是默认被索引的。也就是说，每个字段专门有一个反向索引用于快速检索。而且，与其他数据库不同，它可以在同一个查询中利用所有的这些反向索引，以惊人的速度返回结果。

(1) 文档

程序中大多数的实体或对象能够被序列化为包含键值对的 JSON 对象，键（Key）是字段（Field）或属性（Property）的名字，值（Value）可以是字符串、数字、布尔类型、另一个对象、值数组或者其他特殊类型，例如表示日期的字符串或者表示地理位置的对象。

```
{
  "name": "John Smith",
  "age": 42,
  "confirmed": true,
  "join_date": "2014-06-01",
  "home": {
    "lat": 51.5,
    "lon": 0.1
  },
  "accounts": [
    {
      "type": "facebook",
      "id": "johnsmith"
    },
    {
      "type": "twitter",
      "id": "johnsmith"
    }
  ]
}
```

通常我们可以认为对象（Object）和文档（Document）是等价相通的。不过，它们还是有所差别：对象（Object）是一个 JSON 结构体——类似于哈希、hashmap、字典或者关联数组；对象（Object）中还可能包含其他对象（Object）。在 Elasticsearch 中，文档（Document）这个

术语有着特殊含义。它特指最顶层结构或者根对象（Root Object）序列化成的 JSON 数据（以唯一 ID 标识并存储于 Elasticsearch 中）。

当然，一个文档不只有数据，它还包含元数据（Metadata）——关于文档的信息。三个必须的元数据节点如表 5-7 所示。

表 5-7 文档元数据

节 点	说 明
<code>_index</code>	文档存储的地方
<code>_type</code>	文档代表的对象的类
<code>_id</code>	文档的唯一标识

`_index`：索引（Index）类似于关系型数据库中的“数据库”——它是我们存储和索引关联数据的地方。事实上，我们的数据被存储和索引在分片（Shards）中，索引只是一个把一个或多个分片分组在一起的逻辑空间。对于程序而言，文档只需要存储在索引（Index）中，剩下的细节由 Elasticsearch 处理，选择一个索引名，这个名字必须是全部小写，不能以下划线开头，不能包含逗号。

`_type`：在应用中，我们使用对象表示一些“事物”，例如一个用户、一篇微博、一个评论，或者一封邮件。每个对象都属于一个类（Class），这个类定义了属性或与对象关联的数据。`user` 类的对象可能包含姓名、性别、年龄和 E-mail 地址。在关系型数据库中，我们经常将相同类的对象存储在一个表里，因为它们有着相同的结构。同理，在 Elasticsearch 中，我们使用相同类型（Type）的文档表示相同的“事物”，因为它们的数据结构也是相同的。每个类型（Type）都有自己的映射（Mapping）或者结构定义，就像传统数据库表中的列一样。所有类型下的文档被存储在同一个索引下，但是类型的映射（Mapping）会告诉 Elasticsearch 不同的文档如何被索引。`_type` 的名字可以是大写或小写，不能包含下划线或逗号。

`_id`：`id` 仅仅是一个字符串，它与 `_index` 和 `_type` 组合时，就可以在 Elasticsearch 中唯一标识一个文档。当创建一个文档，你可以自定义 `_id`，也可以让 Elasticsearch 帮你自动生成。

(2) 索引

文档通过 `index` API 被索引——使数据可以被存储和搜索。但是首先我们需要决定文档所在。正如我们讨论的，文档通过其 `_index`、`_type`、`_id` 唯一确定。我们可以自己提供一个 `_id`，或者也可以使用 `index` API 为我们生成一个。

如果你的文档有自然的标识符（例如 `user_account` 字段或者其他值表示文档），你就可以提供自己的 `_id`，使用这种形式的 `index` API：

```
PUT /{index}/{type}/{id}
{
```

```

    "field": "value",
    ...
}

```

比如，我们的索引叫做“website”，类型叫做“blog”，我们选择的ID是“123”，那么这个索引请求就像这样：

```

PUT /website/blog/123
{
  "title": "My first blog entry",
  "text": "Just trying this out...",
  "date": "2014/01/01"
}

```

Elasticsearch 的响应：

```

{
  "_index": "website",
  "_type": "blog",
  "_id": "123",
  "_version": 1,
  "created": true
}

```

响应指出请求的索引已经被成功创建，这个索引中包含 `_index`、`_type` 和 `_id` 元数据，以及一个新元素：`_version`。如果我们的数据没有自然 ID，我们可以让 Elasticsearch 自动为我们生成。请求结构发生了变化：PUT 方法——“在这个 URL 中存储文档”变成了 POST 方法——“在这个文档下存储文档”，原来是把文档存储到某个 ID 对应的空间，现在是把这个文档添加到某个 `_type` 下。

URL 现在只包含 `_index` 和 `_type` 两个字段：

```

POST /website/blog/
{
  "title": "My second blog entry",
  "text": "Still trying this out...",
  "date": "2014/01/01"
}

```

响应内容与刚才类似，只有 `_id` 字段变成了自动生成的值：

```

{
  "_index": "website",
  "_type": "blog",
  "_id": "wM0OSFhDQXGZAWDf0-drSA",
  "_version": 1,
  "created": true
}

```

自动生成的 ID 有 22 个字符长，URL-safe, Base64-encoded string universally unique identifiers,

或者叫 UUIDs。

(3) 数据操作

操作一：检索文档。

想要从 Elasticsearch 中获取文档，我们使用同样的 `_index`、`_type`、`_id`，但是 HTTP 方法改为 GET：GET/website/blog/123?pretty。

在任意的查询字符串中增加 `pretty` 参数，会让 Elasticsearch 美化输出 (pretty-print) JSON 响应以便更加容易阅读。`_source` 字段不会被美化，它的样子与我们输入的一致。

GET 请求返回的响应内容包括 `{"found": true}`。这意味着文档已经找到。如果我们请求一个不存在的文档，依旧会得到一个 JSON，不过 `found` 值变成了 `false`。响应包含了现在熟悉的元数据节点，增加了 `_source` 字段，它包含了在创建索引时我们发送给 Elasticsearch 的原始文档。

```
{
  "_index" : "website",
  "_type" : "blog",
  "_id" : "123",
  "_version" : 1,
  "found" : true,
  "_source" : {
    "title": "My first blog entry",
    "text": "Just trying this out...",
    "date": "2014/01/01"
  }
}
```

此外，HTTP 响应状态码也会变成“404 Not Found”代替“200 OK”。我们可以在 curl 后加 `-i` 参数得到响应头：curl-i-XGET http://localhost:9200/website/blog/124?pretty，现在响应类似于这样：

```
HTTP/1.1 404 Not Found
Content-Type: application/json; charset=UTF-8
Content-Length: 83

{
  "_index" : "website",
  "_type" : "blog",
  "_id" : "124",
  "found" : false
}
```

操作二：检查文档是否存在。

如果你想做的只是检查文档是否存在——你对内容完全不感兴趣——使用 HEAD 方法来代替 GET。HEAD 请求不会返回响应体，只有 HTTP 头：curl-i-XHEAD http://localhost:9200/website/blog/123，如果文档存在，Elasticsearch 将会返回 200 OK 状态：

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=UTF-8
Content-Length: 0
```

如果文档不存在，返回 404 Not Found:

```
HTTP/1.1 404 Not Found
Content-Type: text/plain; charset=UTF-8
Content-Length: 0
```

当然，这只表示你在查询的那一刻文档不存在，但并不表示几毫秒后依旧不存在，因为另一个进程在这期间可能创建新文档。

操作三：创建新文档。

当索引一个文档，我们如何确定是完全创建了一个新的还是覆盖了一个已经存在的呢？请记住 `_index`、`_type`、`_id` 三者唯一确定一个文档。所以要想保证文档是新加入的，最简单的方式是使用 POST 方法让 Elasticsearch 自动生成唯一 `_id`：

```
POST /website/blog/
{ ... }
```

然而，如果想使用自定义的 `_id`，我们必须告诉 Elasticsearch 应该在 `_index`、`_type`、`_id` 三者都不同时才接受请求。为了做到这点有两种方法，它们其实做的是同一件事情。你可以选择适合自己的方式：

1) 第一种方法是使用 `op_type` 查询参数：

```
PUT /website/blog/123?op_type=create
{ ... }
```

2) 第二种方法是在 URL 后加 `/_create` 作为端点：

```
PUT /website/blog/123/_create
{ ... }
```

如果请求成功地创建了一个新文档，Elasticsearch 将返回正常的元数据且响应状态码是 201 Created。另外，如果包含相同的 `_index`、`_type` 和 `_id` 的文档已经存在，Elasticsearch 将返回 409 Conflict 响应状态码，错误信息类似如下：

```
{
  "error" : "DocumentAlreadyExistsException[[website][4] [blog][123]:
    document already exists]",
  "status" : 409
}
```

操作四：更新文档。

文档在 Elasticsearch 中是不可变的——我们不能修改它们。如果需要更新已存在的文档，我们可以使用 index API 重建索引 (Reindex) 或者替换掉它。

```
PUT /website/blog/123
{
  "title": "My first blog entry",
  "text": "I am starting to get the hang of this...",
  "date": "2014/01/02"
}
```

在响应中，我们可以看到 Elasticsearch 把 `_version` 增加了。

```
{
  "_index" : "website",
  "_type" : "blog",
  "_id" : "123",
  "_version" : 2,
  "created": false
}
```

`created` 标识为 `false`，这是因为同索引、同类型下已经存在同 ID 的文档。同时在内部，Elasticsearch 已经标记旧文档为删除并添加了一个完整的新文档。旧版本文档不会立即消失，但你也不能去访问它。Elasticsearch 会在你继续索引更多数据时清理被删除的文档。

同时，Elasticsearch 的另一个 `update` API 会允许修改文档的局部，但事实上遵循与之前 `index` API 完全相同的过程，这个过程如下：

- 1) 从旧文档中检索 JSON。
- 2) 修改它。
- 3) 删除旧文档。
- 4) 索引新文档。

唯一的不同是，`update` API 完成这一过程只需要一个客户端请求即可，不再需要 `get` 和 `index` 请求了。

操作五：删除文档。

删除文档的语法模式与之前基本一致，只不过要使用 `DELETE` 方法：`DELETE/website/blog/123`，如果文档被找到，Elasticsearch 将返回 200 OK 状态码和以下响应体。注意，`_version` 数字已经增加了。

```
{
  "found" : true,
  "_index" : "website",
  "_type" : "blog",
  "_id" : "123",
  "_version" : 3
}
```

如果文档未找到，我们将得到一个 404 Not Found 状态码，响应体是这样的：

```
{
  "found" : false,
```



```

    "_index" : "website",
    "_type" : "blog",
    "_id" : "123",
    "_version" : 4
  }

```

尽管文档不存在——"found" 的值是 false——_version 依旧增加了。这是内部记录的一部分，它确保在多节点间不同操作可以有正确的顺序。

操作六：批量操作 (mget)。

就像 mget 允许我们一次性检索多个文档一样，bulk API 允许我们使用单一请求来实现多个文档的 create、index、update 或 delete。这对索引类似于日志活动这样的数据流非常有用，它们可以以成百上千的数据为一个批次按序进行索引。bulk 请求体如下，它有一点不同寻常：

```

{ action: { metadata } }\n
{ request body      }\n
{ action: { metadata } }\n
{ request body      }\n
...

```

这种格式类似于用 "\n" 符号连接起来的一行一行的 JSON 文档流 (stream)。两个重要的点需要注意：

- 1) 每行必须以 "\n" 符号结尾，包括最后一行。这些都是作为每行有效的分离而做的标记。
- 2) 每一行的数据不能包含未被转义的换行符，它们会干扰分析——这意味着 JSON 不能被美化打印。

action/metadata 这一行定义了文档行为 (what action) 发生在哪个文档 (which document) 之上。行为 (action) 必须是以下几种，如表 5-8 所示。

表 5-8 文档行为定义

行 为	解 释
create	当文档不存在时创建之
index	创建新文档或替换已有文档
update	局部更新文档
delete	删除一个文档

在索引、创建、更新或删除时必须指定文档的 _index、_type、_id 这些元数据 (metadata)。比如，删除请求看起来像这样：

```

{ "delete": { "_index": "website", "_type": "blog", "_id": "123" }}

```

请求体 (request body) 由文档的 _source 组成——文档所包含的一些字段以及其值。它被 index 和 create 操作所必需，这是有道理的：你必须提供文档用来索引。这些还被 update 操作所必需，而且请求体的组成应该与 update API (doc、upsert、script 等) 一致。删除操作不需要请求体 (request body)。

```
{ "create": { "_index": "website", "_type": "blog", "_id": "123" }}
{ "title": "My first blog post" }
```

如果定义 `_id`, ID 将会被自动创建:

```
{ "index": { "_index": "website", "_type": "blog" }}
{ "title": "My second blog post" }
```

为了将这些放在一起, `bulk` 请求表单是这样的:

```
POST /_bulk
{ "delete": { "_index": "website", "_type": "blog", "_id": "123" }}
{ "create": { "_index": "website", "_type": "blog", "_id": "123" }}
{ "title": "My first blog post" }
{ "index": { "_index": "website", "_type": "blog" }}
{ "title": "My second blog post" }
{ "update": { "_index": "website", "_type": "blog", "_id": "123", "_retry_on_
conflict" : 3} }
{ "doc" : { "title" : "My updated blog post" } }
```

Elasticsearch 响应包含一个 `items` 数组, 它罗列了每一个请求的结果, 结果的顺序与我们请求的顺序相同:

```
{
  "took": 4,
  "errors": false,
  "items": [
    { "delete": {
      "_index": "website",
      "_type": "blog",
      "_id": "123",
      "_version": 2,
      "status": 200,
      "found": true
    }},
    { "create": {
      "_index": "website",
      "_type": "blog",
      "_id": "123",
      "_version": 3,
      "status": 201
    }},
    { "create": {
      "_index": "website",
      "_type": "blog",
      "_id": "EiwfApScQiyy7TIKFxRCTw",
      "_version": 1,
      "status": 201
    }},
    { "update": {
      "_index": "website",
```

```

    "_type": "blog",
    "_id": "123",
    "_version": 4,
    "status": 200
  }}
]
}}

```

每个子请求都被独立地执行，所以一个子请求的错误并不影响其他请求。如果任何一个请求失败，顶层的 `error` 标记将被设置为 `true`，然后错误的细节将在相应的请求中被报告：

```

POST /_bulk
{ "create": { "_index": "website", "_type": "blog", "_id": "123" }}
{ "title": "Cannot create - it already exists" }
{ "index": { "_index": "website", "_type": "blog", "_id": "123" }}
{ "title": "But we can update it" }

```

响应中我们将看到 `create` 文档 123 失败了，因为文档已经存在，但是后来在 123 上执行的 `index` 请求成功了：

```

{
  "took": 3,
  "errors": true,
  "items": [
    { "create": {
      "_index": "website",
      "_type": "blog",
      "_id": "123",
      "status": 409,
      "error": "DocumentAlreadyExistsException
        [[website][4] [blog][123]:
        document already exists]"
    }},
    { "index": {
      "_index": "website",
      "_type": "blog",
      "_id": "123",
      "_version": 5,
      "status": 200
    }}
  ]
}

```

其中，`"errors": true` 代表一个或多个请求失败。`"error": DocumentAlreadyExists Exception` 错误消息说明了什么请求错误。`"status": 200` 代表第二个请求成功了，状态码是 200 OK。这些说明 `bulk` 请求不是原子操作——它们不能实现事务。每个请求操作是分开的，所以每个请求的成功与否不干扰其他操作。

整个批量请求需要被加载到接受请求节点的内存中，所以请求越大，给其他请求可用的

内存就越小。这样会有一个最佳的 bulk 请求大小，超过这个大小，性能不再提升而且可能降低。最佳大小，当然并不是一个固定的数字。它完全取决于你的硬件、你文档的大小和复杂度以及索引和搜索的负载。幸运的是，这个最佳点（sweet spot）还是容易找到的：

试着批量索引标准的文档，随着大小的增长，当性能开始降低，说明你每个批次的大小太大了。开始的数量可以在 1000~5000 个文档，如果你的文档非常大，可以使用较小的批次。1000 个 1KB 的文档和 1000 个 1MB 的文档大不相同，一个好的批次最好保持在 5MB ~ 15MB 大小。

4. 中文分词

中文分词是全文检索和文本挖掘的基础，作为自然语言处理的第一工序，有着十分重要的意义，在一定程度上可以说没有分词，所谓的分类、句法树、聚类、特征词提取、文本摘要等都是空谈，对于输入的一段中文，成功地进行中文分词，可以达到电脑自动识别语句含义的效果。中文分词技术属于自然语言处理技术范畴，对于一句话，人可以通过自己的知识来明白哪些是词，哪些不是词，但如何让计算机也能理解？其处理过程就是分词算法。

存在中文分词技术，是由于中文在基本文法上有其特殊性，具体表现在：与以英文为代表的拉丁语系语言相比，英文以空格作为天然的分隔符，而中文由于继承自古代汉语的传统，词语之间没有分隔。在中文里“词”和“词组”边界模糊，现代汉语的基本表达单元虽然为“词”，且以双字或者多字词居多，但由于人们认识水平的不同，对词和短语的边界很难区分。比如，“对随地吐痰者给予处罚”，“随地吐痰者”本身是一个词还是一个短语，不同的人会有不同的标准，即使是同一个人也可能做出不同判断。

分词用简单直白的方法处理就是建立一个自己的词库，然后用正向或逆向方式遍历句子，发现有在词库中的就进行切词，这样就能完成一个简单的分词程序。当然效率方面就有很多改进方式了，例如词库的存储结构，程序是否执行单例模式，查询算法之类都会直接影响速率，这方面不再做过多讲解了。同时，中文是一种十分复杂的语言，让计算机理解中文语言更是困难。所以在中文分词过程中，有两大难题一直没有完全突破。

歧义识别

歧义是指同样的一句话，可能有两种或者更多的切分方法。分词过程中歧义产生的根源可归纳为以下三类：

- 由自然语言的二义性所引起的歧义。
- 由机器自动分词产生的特有歧义。
- 由于分词词典的大小引起的歧义。

由自然语言的二义性所引起的歧义，称为第一类歧义。比如，“乒乓球拍卖完了”可切分为“乒乓球/拍卖/完了”又可以切分为“乒乓球拍/卖/完了”。这两种切分形式无论在语法上还是语义上都是正确的，就是人工分词也会产生歧义，只有结合上下文才能给出正确的切分。

由机器自动分词产生的特有歧义，称为第二类歧义。比如，“在这种环境下工作是太可怕了”用机器切分，可以切分为“在/这种/环境/下工/是/太/可怕/了”，也可以切分为

“在 / 这种 / 环境 / 下 / 工作 / 是 / 太 / 可怕 / 了”。对本句来说,只有第二种切分是正确的。用人工分词是不可能产生歧义的,歧义是由于机器机械切分产生的。

由于分词词典的大小引起的歧义,称为第三类歧义。例如,“王小二是一个农民”用机器切分被分为“王 / 小 / 二 / 是 / 一个 / 农民”,这里“王小二”是一个人名,在汉语中应是一个词,所以这个切分是错误的。由于机器自动分词是根据分词词典进行的,故词典中没有的词,就不可能被正确切分,分词词典不可能也没有必要包括所有的词(例如人名、地名),同时,词典中所包括的词越多,就会产生新的歧义。比如,“发展社会主义的新乡村”新乡是一个地名,若词典中有该词,则“新乡村”是一个歧义字段。因此,不论词典的大与小都可能产生歧义。

经过统计表明,第一类歧义字段只占歧义字段总数的5%左右,剩下的就都是第二类歧义字段和第三类歧义字段。故自动分词阶段对歧义的研究主要集中在对第二类、第三类歧义字段的研究。而对于第二类歧义,又主要有两种:组合型歧义与交集型歧义。组合型歧义就是对于字符串AB,可以切分为AB,又可以切分为A/B;交集型歧义就是对于字符串ABC,可以切分为AB/C,又可以切分为A/BC。这其中交集型歧义又占了绝大多数,据统计达94%,因此处理好交集型歧义在汉语分词中有着非常重要的地位。

新词识别

命名实体(人名、地名)、新词,专业术语称为未登录词,也就是那些在分词词库中没有收录,但又确实能称为词的那些词。最典型的是人名,人们可以很容易地理解。句子“王军虎去广州了”中,“王军虎”是个词,因为是一个人的名字,但要是让计算机去识别就困难了。如果把“王军虎”作为一个词收录到词库中,全世界有那么多名字,而且每时每刻都有新增的人名,收录这些人本身就是一项既不划算又巨大的工程。即使这项工作可以完成,还是会存在问题,例如在句子“王军虎头虎脑的”中,“王军虎”还能不能算词?

由于我们是自己建立的词库,所以对词库的维护是很重要的,互联网每天都会产生一些新词,这时如果全由人工进行维护,则成本是相对较高的且在准确及时性上有待考量。这时可以考虑用程序的方式进行事先筛选和提取,然后以人工做校正。在程序方面有三种做法:

(1) 方法一:信息熵

采用信息熵的方式来度量两个词之间的紧密程度。将紧密程度高,而又不在我们词库中的词标识出来。比如,一篇文章里有“新长城办公室”(实际意思为:新长城/办公室)六个字,而我们词库里只有“新”“长城”“办公室”。则这段文字将会被切成:新/长城/办公室,而这样的分词明显是不符合实际的。采用信息熵后可以解决这个问题,信息熵的一般公式为

$$I(X, Y) = \log_2 \frac{P(X, Y)}{P(X)P(Y)}$$

解释起来其实也挺简单,就是如果X和Y的出现相互独立,则 $P(X, Y)$ 的值和 $P(X)P(Y)$ 的值相等,因此 $I(X, Y)$ 为0。如果X和Y密切相关, $P(X, Y)$ 将比 $P(X)P(Y)$ 大很多, $I(X, Y)$ 的值也就远大于0。如果X和Y几乎不会相邻出现,而它们各自出现的概率又比较大,那么 $I(X, Y)$ 将取负值,这时候X和Y负相关。所以只要算出两个词的相关程度远大于0,我

们设置一个阈值，当大于某个值时就将这两个词作为一个新词标注出来，为提高准确性，人工去校正即可。

（2）方法二：基于搜索日志

从搜索日志里发现新词：这个应该很好理解，就是人们经常搜索什么，把搜索时的词入库检索，然后按查询的频次倒序一排也能有助于我们发现新词。如果我们的系统不提供，当然也可以直接借助现成的搜索引擎的相关搜索去发现了。

（3）方法三：使用输入法收集

利用输入法来收集新词：搜狗输入法的新词维护还是不错的，具体维护方法就不详细描述。

另外，中文分词时，有些词并不是通过词库匹配出来的，而是根据一定的规则自动识别的。常见的如数字、日期、邮箱、电话号码等，当然可能有人说这直接用正则表达式就可以。确实是这样的，正则表达式完全可以实现，同时也是一种简单的处理方法，只是相比扩展性之类而言，可能还是有限状态机更好些。

有限状态机简称状态机，是表示有限个状态以及在这些状态之间的转移和动作等行为的数学模型，是由状态（State）、变换（Transition）和行动（Action）组成的行为模型。有限状态机首先在初始状态（Start State），接收输入事件（Input Event）后转移到下一个状态。有限状态机通过条件（Guard）判断是否符合转移条件。具体的代码实现由于侧重点关系在这里不做过多讲解。以下介绍几款主流的中文分词器：

IKAnalyzer

IKAnalyzer 是一个开源的，基于 Java 语言开发的轻量级的中文分词工具包。从 2006 年 12 月推出 1.0 版开始，IKAnalyzer 已经推出了 3 个大版本。最初，它是以开源项目 Luence 为应用主体的，结合词典分词和文法分析算法的中文分词组件。新版本的 IKAnalyzer3.X 则发展为面向 Java 的公用分词组件，独立于 Lucene 项目，同时提供了对 Lucene 的默认优化实现。

IKAnalyzer3.X 的特性如下：

- ❑ 采用了特有的“正向迭代最细粒度切分算法”，具有 60 万字 / 秒的高速处理能力。
- ❑ 采用了多子处理器分析模式，支持：英文字母（IP 地址、E-mail、URL）、数字（日期、常用中文数量词、罗马数字、科学计数法）、中文词汇（姓名、地名处理）等分词处理。
- ❑ 优化的词典存储，更小的内存占用。支持用户词典扩展定义。
- ❑ 针对 Lucene 全文检索优化的查询分析器 IKQueryParser（笔者吐血推荐）；采用歧义分析算法优化查询关键字的搜索排列组合，能极大地提高 Lucene 检索的命中率。

分词效果示例如下：

（1）示例一

文本原文 1: IKAnalyzer 是一个开源的，基于 Java 语言开发的轻量级的中文分词工具包。从 2006 年 12 月推出 1.0 版开始，IKAnalyzer 已经推出了 3 个大版本。

分词结果: IKAnalyzer| 是 | 一个 | 一 | 个 | 开源 | 的 | 基于 | Java | 语言 | 开发 | 的 | 轻量级 | 量

级|的|中文|分词|工具包|工具|从|2006|年|12|月|推出|1.0|版|开始|IKAnalyzer|已经|推出|出了|3|个大|个|版本。

(2) 示例二

文本原文 2: 永和服装饰品有限公司。

分词结果: 永和|和服|服装|装饰品|装饰|饰品|有限|公司。

(3) 示例三

文本原文 3: 作者博客: linliangyi2007.javaeye.com 电子邮件: linliangyi2005@gmail.com。

分词结果: 作者|博客|linliangyi2007.javaeye.com|linliangyi|2007|javaeye|com|电子邮件|邮件地址|linliangyi2005@gmail.com|linliangyi|2005|gmail|com。

与相关组件的版本兼容如图 5-62 所示。

IK 分词器版本	Lucene 版本	Solr 版本
3.1.3GA 及先前版	兼容 2.9.1 及先前版本	没有 Solr 接口
3.1.5GA	兼容 2.9.1 及先前版本	对 Solr1.3 提供接口实现 (详细请参考对应版本使用手册)
3.1.6GA	兼容 2.9.1 及先前版本	对 Solr1.3、Solr1.4 提供接口实现 (详细请参考对应版本使用手册)
3.2.0GA	兼容 Lucene2.9 及 3.0 版本 不支持 Lucene2.4 及先前版本	仅对 Solr1.4 提供接口实现 (请参考本手册 Solr 部分说明)

图 5-62 IK 分词的不同版本与 Lucene 和 Solr 的兼容性

Paoding

庖丁中文分词库是一个使用 Java 开发的, 可结合到 Lucene 应用中的, 为互联网、企业内部网使用的中文搜索引擎分词组件。Paoding 填补了国内在中文分词方面开源组件的空白, 致力于此并希冀成为互联网网站首选的中文分词开源组件。Paoding 中文分词追求分词的高效率和用户良好体验。

Paoding's Knives 中文分词具有极高的效率和高扩展性。引入隐喻, 采用完全的面向对象设计, 构思先进。支持不限制个数的用户自定义词库, 纯文本格式, 一行一词, 使用后台线程检测词库的更新, 自动编译更新过的词库到二进制版本, 并加载。比较痛苦的是, Paoding 中文分词几乎没有介绍文档, 代码里有一些注释, 但因为实现比较复杂, 读代码还是有一些难度的。

高效率: 在 PIII 1G 内存个人机器上, 1 秒可准确分词 100 万汉字。

采用基于不限制个数的词典文件对文章进行有效切分, 能够对词汇分类进行定义, 能够对未知的词汇进行合理解析, 如图 5-63 所示为示例代码。

mmseg4j

mmseg4j 是用 Chih-Hao Tsai 的 MMSeg 算法实现的中文分词器, 并实现 Lucene 的 analyzer 和 Solr 的 TokenizerFactory 以方便在 Lucene 和 Solr 中使用。MMSeg 算法有两种分词方法: Simple 和 Complex, 都是基于正向最大匹配。Complex 加了四个规则过虑。官方说: 词语的正确识

别率达到了 98.41%。mmseg4j 已经实现了这两种分词算法。详细介绍如下：

```

1 //生成analyzer实例
2 Analyzer analyzer = new PaodingAnalyzer(properties);
3 //取得Token流
4 TokenStream stream = analyzer.tokenStream("", reader);
5
6 //重置到流的开始位置
7 stream.reset();
8
9 //添加工具类
10 TermAttribute termAtt = (TermAttribute) stream.addAttribute(TermAttribute.class);
11 OffsetAttribute offAtt = (OffsetAttribute) stream.addAttribute(OffsetAttribute.class);
12
13 //循环打印所有分词及其位置
14 while (stream.incrementToken()) {
15     System.out.println(termAtt.term() + " " + offAtt.startOffset() + " " + offAtt.endOffset());
16 }

```

图 5-63 庖丁中文分词应用示例

- ❑ 在 com.chenlb.mmseg4j.example 包里的类示例了三种分词效果。
- ❑ 在 com.chenlb.mmseg4j.analysis 包里扩展 lucene analyzer。MMSegAnalyzer 默认使用 max-word 方式分词（还有 ComplexAnalyzer、SimplexAnalyzer、MaxWordAnalyzer）。
- ❑ 在 com.chenlb.mmseg4j.solr 包里扩展 solr tokenizerFactory。dicPath 指定词库位置（每个 MMSegTokenizerFactory 可以指定不同的目录，当为相对目录时，是相对 solr.home 的目录），mode 指定分词模式（simple|complex|max-word，默认是 max-word）。
- ❑ 词典用 mmseg.dic.path 属性指定或在当前目录下的 data 目录，默认是 ./data 目录。
- ❑ java-Dmmseg.dic.path=./data-jar mmseg4j-1.6.jar 这里是字符串。
- ❑ java-cp.; mmseg4j-1.6.jar com.chenlb.mmseg4j.example.Simple 这里是字符串。
- ❑ java-cp.; mmseg4j-1.6.jar com.chenlb.mmseg4j.example.MaxWord 这里是字符串。
- ❑ 一些字符的处理英文、俄文、希腊、数字（包括①(-)1.）的分出一连串的。目前版本没有处理小数字问题，例如 I II III 是单字分，字库（chars.dic）中没找到也单字分。词库（强制使用 UTF-8）：

- ❑ data/chars.dic 是单字与语料中的频率，一般不用改动，1.5 版本中已经加到 mmseg4j 的 jar 里，我们不需要关心它，当然你在词库目录放这个文件可能覆盖它。
- ❑ data/units.dic 是单字的单位，默认读 jar 包里的，你也可以自定义覆盖它，该功能是试行，如果不喜欢它，可以用空的 units.dic 文件覆盖它。
- ❑ data/words.dic 是词库文件，一行一词，当然你也可以使用自己的，1.5 版本使用 sogou 词库，1.0 版本是用 rmmseg 带的词库。
- ❑ data/wordsxxx.dic 1.6 版支持多个词库文件，在 data 目录（或你定义的目录）下读到“words”前缀且“.dic”为后缀的文件，例如 data/words-my.dic。

Maven 配置如图 5-64 所示。

分词速度：

- ❑ 1.5 版的分词速度，simple 算法是 1100kbit/s 左右、complex 算法是 700kbit/s 左右（测试机：AMD athlon 64 2800+1G 内

```

1 <dependency>
2   <groupId>com.chenlb.mmseg4j</groupId>
3   <artifactId>mmseg4j-core</artifactId>
4   <version>1.10.0</version>
5 </dependency>

```

图 5-64 Maven 配置示例

存 xp)。

- ❑ 1.6 版在 complex 基础上实现了最多分词 (max-word)。“很好听”->“很好|好听”;“中华人民共和国”->“中华|华人|共和|国”;“中国人民银行”->“中国|人民|银行”。
- ❑ 1.7-beta 版, 目前 complex 算法是 1200kbit/s 左右, simple 算法是 1900kbit/s 左右, 但内存开销了 50M 左右, 上几版都是在 10M 左右。

盘古分词

盘古分词是一个基于 .net framework 的中英文分词组件。主要功能如下:

- ❑ 中文未登录词识别: 盘古分词可以对一些不在字典中的未登录词自动识别。
- ❑ 词频优先: 盘古分词可以根据词频来解决分词的歧义问题。
- ❑ 多元分词: 盘古分词提供多重输出解决分词粒度和分词精度权衡的问题。
- ❑ 中文人名识别: 输入:“张三说的确实在理”, 分词结果: 张三/说/的/确实/在理/。
输入“李三买了一张三角桌子”, 分词结果: 李三/买/了/一张/三角/桌子/。
- ❑ 强制一元分词: 输入“张三说的确实在理”, 分词结果: 张(0,1)/张三(0,5)/三说的(1,1)/三(1,1)/说(2,5)/的(3,5)/确(4,1)/确实(4,5)/实(5,1)/在(6,1)/在理(6,5)/理(7,1)/。
- ❑ 繁体中文分词: 输入“我的選擇”, 分词结果: 我/的/選擇/。
- ❑ 同时输出简体和繁体: 输入“我的選擇”, 分词结果: 我(0,5)/的(1,5)/选择(2,1)/選擇(2,5)/。
- ❑ 中文词性输出: 盘古分词可以将已登录词的中文词性输出给用户, 以方便用户做进一步处理。
- ❑ 全角字符支持: 盘古分词可以识别全角的字母和数字。

(1) 英文分词

英文分词: 英文单词通常都是靠空格等符号分割, 这个比较简单, 盘古分词分英文自然也没有什么问题。英文分词包括以下三种功能:

- ❑ 英文专用词识别: 一些英文简写是字母符号混合, 或者是字母数字混合, 分词起来就不能按照空格符号这样分割了, 对于字母符号混合的如 U.S.A, 只要将这个词录入到字典中, 盘古分词就可以分出整词。对于字母和数字混合的, 盘古分词会自动作为整词输出。
- ❑ 英文原词输出。
- ❑ 英文大小写同时输出。

(2) 其他功能

- ❑ 停用词过滤: 对于一些标点符号、连词、助词等有时需要在分词时过滤掉, 盘古分词提供一个 StopWord.txt 文件, 用户只要将需要过滤的词加入到这个文件中, 并将停用词过滤开发打开, 就可以过滤掉这些词。
- ❑ 设置分词权值: 盘古分词可以让用户对如下特性设置自定义权值: ①未登录词权值;

②最匹配词权值；③次匹配词权值；④再次匹配词权值；⑤强行输出的单字的权值；⑥数字的权值；⑦英文词汇权值；⑧符号的权值；⑨强制同时输出简繁体汉字时，非原来文本的汉字输出权值。

- 字典管理：盘古分词提供一个字典管理工具 DictManage，通过该工具，你可以增加、修改和删除字典中的单词。
- 动态加载字典：通过字典工具增加、修改和删除字典中的单词后，盘古分词会自动将新的字典文件加载进去，而不需要重新启动。
- 关键词高亮组件：Lucene 提供了一个关键词高亮组件，但这个组件对中文的支持不是特别好，特别是如果还有多元分词的情况，处理的就更不好。盘古分词提供了一个针对中文和英文的关键词高亮组件 PanGu.HighLight，其对中文的支持要好于 Lucene 那个高亮组件。

(3) 性能指标

Core Duo 1.8 GHz 下单线程分词速度为 390K 字符每秒，2 线程分词速度为 690K 字符每秒。

5.2 相关技术

5.2.1 数据可视化

数据可视化技术包括 JavaScript、ActionScript、CSS、xHTML 等“传统”技术与 Adobe RIA、Google Gears，以及概念性较强的交互式设计、艺术性较强的视觉设计等。

它涵盖的领域包括可用性工程、交互设计、软件工程等。根据该词可以做这样的理解，用互联网来做比喻，凡是通过浏览器到用户端计算机的统称为前端技术，相反存储于服务器端的统称为后端技术。

1. HTML5

HTML5 是万维网的核心语言、超文本标记语言（HTML）的第五次重大修改，已经于 2014 年 10 月正式定稿。目前，大部分现代浏览器已经具备了某些 HTML5 支持。HTML5 是 Web 时代最前沿的技术，它特有的 canvas 标签和多种选择的游戏开发引擎，让游戏开发更便捷。如果说苹果重新发明了手机，那么 HTML5 则重新定义了网络。它是连接手机、平板电脑、PC 以及其他移动终端的桥梁，可以更丰富地展现页面，让视频、音频、游戏以及其他元素构成一场华丽的代码盛宴。

HTML5 是 Web 的未来，不仅在电脑端，而且在移动端也一定会得到广泛的应用。据统计，2013 年全球有 10 亿手机浏览器支持 HTML5，同时 HTML Web 开发者数量达到 200 万。毫无疑问，HTML5 将成为未来 5 ~ 10 年内，移动互联网领域的主宰者。据 IDC 的调查报告统计，截至 2012 年 5 月，有 79% 的移动开发商已经决定要在其应有程序中整合 HTML5 技术。Web 技术发展越来越迅速，HTML5 的到来更是把 Web 技术推向了巅峰，目前 HTML5

技术已经日趋成熟, HTML5 的未来十分光明, 值得我们去学习。

除了实现之前 HTML 可以实现的功能外, HTML5 还可以做以下特别的事情:

- ❑ 本地存储基于 HTML5 开发的网页 APP 拥有更短的启动时间和更快的联网速度, 这些全得益于 HTML5 APP Cache, 以及本地存储功能。
- ❑ 实现多媒体更加简单: 利用 HTML5 的 <video> 和 <audio> 可以非常方便地在网页上添加视频和音频, 不需要很复杂的代码, 就能打造一款功能齐全的 HTML5 播放器。
- ❑ 三维图形和动画 HTML5 的 3D 引擎可以更方便地实现 3D 效果, 而且更加逼真。
- ❑ CSS3 的运用: CSS3 提供更多的 CSS 属性, 可以做更丰富的渲染效果。

HTML5 优势

- ❑ 一次编写, 随处部署: HTML5 可以在多种设备上运行, 这是其他方式都无法做到的。
- ❑ 在互联网中分享: HTML5 应用都有一个 URL, 因此它可以在互联网中被随意分享, 并且在搜索时即可被找到。
- ❑ 多厂商标准, 建立在协议之上: HTML5 是众多公司努力的结果, 没有一家厂商可以左右它的方向。
- ❑ 适用于多种环境: HTML5 应用可以使用交互式设计来提供最佳体验, 而不需要更改代码。你可以从桌面到手机到平板电脑无缝进行切换, 而无需重复安装不同的应用。

HTML5 主要功能

为了更好地处理今天的互联网应用, HTML5 添加了很多新元素及功能, 例如图形绘制、多媒体内容、更好的页面结构、更好的形式处理, 以及 API 拖放元素等。

(1) 脱机功能

- ❑ HTML5 透过 JavaScript 提供了数种不同的脱机储存功能, 相对于传统的 Cookie 而言有更好的弹性以及架构, 并且可以储存更多的内容。
- ❑ WebStorage: 比 Cookies 更大、更有弹性的储存。
- ❑ Web SQL Database: 本地端的 SQL 数据库。
- ❑ Indexed DB: Key-value 的本地数据库。
- ❑ Application Cache: 将部分常用的网页内容 Cache 起来。

(2) 实时通信

以往网站由于 HTTP 协议以及浏览器的设计, 实时的互动性相当受限, 只能使用一些技巧来“仿真”实时的通信效果, 但 HTML5 提供了完善的实时通信支持。

- ❑ WebSocket: 实时的 Socket 联机。
- ❑ Web Workers: 以往 JavaScript 都是 single thread, 透过 Worker 可以有多个运算。
- ❑ Notifications: 原生的提示讯息, 类似像 OS X 的 Growl 提示。

(3) 档案以及硬件支持

不知道大家有没有发现, 在 Gmail 等新的网页程序当中, 已经可以透过拖拉的方式将档案作为邮件附件? 这就是这部分 HTML5 档案的功能中的 Dragn Drop 和 File API。

- ❑ Dragn Drop: HTML 元素的拖拉。
- ❑ File API: 读取用户本机计算机的内容。
- ❑ Geolocation: 地理定位。
- ❑ Device orientation: 手持装置的方向。
- ❑ Speech input: 语音输入。

(4) 语义化

语义化的网络是可以让计算机能够更加理解网页的内容，对于如搜索引擎的优化（SEO）或是推荐系统可以有很大的帮助。

- ❑ New tags: 新的标签，例如 header、section 等。
- ❑ Application tags: 也是新的标签，例如 meter、progress 等。
- ❑ Microdata: 加入语义的数据让搜索引擎等网站可以正确显示。
- ❑ Form type: form 可以加入的 type 便多了，包含 email 和 tel 等属性，浏览器会协助进行数据格式的验证。

(5) 多媒体

- ❑ Audio、Video 的卷标支持以及 Canvas 的功能应该是大家对于 HTML5 最熟悉的部分了，这也是许多人认为 Flash 会被取代的主要原因。
- ❑ Audio video: 影片和音乐的原生播放支持。
- ❑ Canvas: 2D 的绘图功能支持。
- ❑ Canvas 3D: 3D 的绘图功能支持。
- ❑ SVG: 向量图支援。

一般广义而言的 HTML5 则包含了 HTML、CSS 和 JavaScript 三个部分，不单单只是 HTML 部分，CSS3 和 JavaScript 也有许多创新，让整个网页程序功能更加缤纷。

CSS3 技术

CSS3 支持了字体的嵌入、版面的排版，以及最令人印象深刻的动画功能。

- ❑ Selector: 更有弹性的选择器。
- ❑ Webfonts: 嵌入式字体。
- ❑ Layout: 多样化的排版选择。
- ❑ Styling radius gradient shadow: 圆角、渐层、阴影。
- ❑ Border background: 边框的背景支持。
- ❑ Transition: 组件的移动效果。
- ❑ Transform: 组件的变形效果。
- ❑ Animation: 将移动和变形加入动画支持。

JavaScript 技术

HTML5 相较于 JavaScript 新增了 DOM 的 API 和浏览器上下页的记录修改。

- ❑ DOM API: 更方便地查询 DOM 组件。

□ History API: 浏览器的上下页内容修改, 方便 AJAX 可以保留浏览记录。

截至目前, 主流的网页浏览器 Firefox 5、Chrome 12 和 Safari 5 都已经支持了许多 HTML5 标准, 而且目前最新版的 IE10 也支持了许多 HTML5 标准, 随着使用者陆续升级到新版的浏览器, 开发者应该在现在就可以着手开发! 而 Modernizer 也是一个相当重要的 JavaScript 函数库, 提供开发者以轻松的方式判别目前使用者的浏览器是否有支持特定的 HTML5 功能。

从可用性的角度上看, HTML5 可以更好地促进用户与网站间的互动情况, 多媒体网站可以获得更多的改进, 特别是在移动平台上的应用, 使用 HTML5 可以提供更多高质量的视频和音频流。

目前, 事实是 iPhone 和 iPad 将不会支持 Flash, 同时 Adobe 公司也在近期公开声明将停止 Flash 基于移动平台的开发, 现在我们已经可以这么说: 移动平台日后的视频音频将是 HTML5 的天下!

2. jQuery

jQuery 是一种轻巧的 JavaScript 框架 (即库)。jQuery 封装了 DOM 的操作, 定义了大量的方法来完成各种复杂的操作, 让程序员简化了开发工作; 优化了对 HTML 的文档操作、事件处理、动画设计和 Ajax 交互等; 解决了不同浏览器之间的差异性。

jQuery 只建立一个名为 jQuery 的对象, 其别名为 \$, 由该对象或其别名调用相关的方法实现复杂的功能。\$(参数) 等价于 jQuery(参数), 其中的参数不仅仅是元素选择器或其他各种选择器, 还可以是方法。如果参数是函数, 即 \$(function(){...})\$, 则实际上是 \$(“document”).ready() 方法的简写, 它也相当于 JavaScript 中的 window.onload=function(){...} 事件处理函数。

jQuery 对象是通过 jQuery 框架包装 DOM 对象之后产生的一个新的对象。从本质上分析, jQuery 仅是 DOM 对象的集合。因此, 我们可以把 DOM 对象看做是一个独立的个体, 而把 jQuery 看成是多个 DOM 对象组成的集合。jQuery 框架为 jQuery 对象定义了独立使用的方法和属性, 它无法直接调用 DOM 对象中的方法和属性, 相反, DOM 对象也无法直接调用 jQuery 对象的方法和属性。

使用 jQuery 框架之后, JavaScript 操作的不再是 HTML 元素对应的 DOM 对象, 而是包装 DOM 对象的 jQuery 对象。JavaScript 通过调用 jQuery 对象的方法来改变它所包含的 DOM 对象的属性, 从而实现动态更新 HTML 页面。由此可见, 使用 jQuery 对象动态更新 HTML 页面只需要两个步骤: ①获取 jQuery 对象。jQuery 对象通常是对 DOM 对象的包装。②调用 jQuery 对象的方法来改变自身。当 jQuery 对象被改变时, jQuery 包装的 DOM 对象随之改变, HTML 页面的内容也随之更新了。

jQuery 对象的很多改变自身属性的方法都有返回值, 就是返回 jQuery 对象本身, 因此可以连续多次调用改变自身属性的方法。jQuery 对象和 DOM 对象可以相互转换, 只不过 jQuery 对象包含了多个 DOM 元素, 而 DOM 对象本身就是个 DOM 元素。因此, jQuery 对象转换为 DOM 时, 需逐个转换 (可通过数组元素或用 get(...) 方法获得一个元素), 如图 5-65

所示。

利用 jQuery 框架来开发时，必须将该框架导入到要开发的项目文件中。因此，只需要将相应的 *.js 文件导入后就可以进行 jQuery 开发了。格式如下：

```
<script src="jQuery 框架文件的位置" type="text/javascript"></script>
```

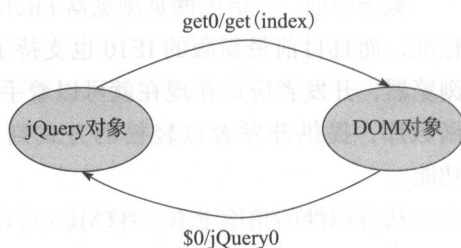


图 5-65 jQuery 请求过程

jQuery 与其他 JavaScript 库共存的问题

jQuery 是 JavaScript 库中的一种，JavaScript 还有其他一些库的支持，例如 Prototype 库。当有多种 JavaScript 库同时使用时，可能会有一些冲突，例如 \$() 在 jQuery 库中表示的是 jQuery 对象，而在 Prototype 库中表示的是 DOM 对象。这时就需要解决这个冲突：jQuery 库提供了 noConflict() 方法用于取消 \$() 的作用，但 jQuery() 方法依然可用。如果使用 jQuery() 时觉得繁琐，可以通过重新为 jQuery 取别名：var lee = jQuery.noConflict(); 这时 jQuery 对象的别名就是 lee，其操作方式和 \$ 一样。

jQuery 对象的核心函数

- ❑ jQuery(expression, [context]): 该函数获取 expression 对应的 DOM 对象包装成的 jQuery 对象。其中 expression 既支持 CSS 选择器，也支持 XPath 语法。由于 expression 表达式可能对应单个 DOM 元素，也可能对应多个 DOM 元素，因此该方法可能返回单个 jQuery 对象，也可能返回 jQuery 对象数组。
- ❑ jQuery(html, [ownerDocument]): 该函数根据 html 参数（该参数是个 HTML 字符串，例如 "<input .../>"）创建一个或多个 DOM 对象，返回包装这些 DOM 对象的 jQuery 对象。其中 ownerDocument 是可选参数，指定使用 ownerDocument（document 对象）来创建 DOM 对象。
- ❑ jQuery(elements): 将一个或多个 DOM 元素包装为 jQuery 对象。elements 既可以是单个的 DOM 对象，也可以是多个 DOM 对象。该方法返回包装这些 DOM 对象的 jQuery 对象。
- ❑ jQuery(callback): 这种用法是 \$(document).ready() 的缩写，其中 callback 制定一个函数，在页面加载完成时 callback 函数被激发。该函数返回页面 document 对象包装成的 jQuery 对象。

HTML 文档加载顺序

- ❑ ①解析 HTML 结构；
- ❑ ②加载外部脚本和样式表文件；
- ❑ ③解析并执行脚本代码；
- ❑ ④构造 HTML DOM 模型；
- ❑ ⑤加载图片等外部文件；

□ ⑥页面加载完毕。

ready 事件和 load 事件的比较

□ ready 事件属于 jQuery; load 事件属于 JavaScript。

□ load 事件必须等到网页中所有内容全部加载完毕后(第⑥步)才被执行; ready 事件是在 DOM 结构绘制完毕(第④步)就开始执行。

□ load 事件只能被编写一次; ready 事件可以被编写多次。

jQuery UI

jQuery UI 的设计一般是通过 jQuery 相关的插件来实现的,所以我们一般需要去下载相关的 jQuery UI 插件。开发时,我们引入相关的 jQuery UI 文件及相关的 CSS 文件即可调用相关的 UI。

3. Highcharts

Highcharts 是一款纯 JavaScript 编写的图表库,功能强大、开源、美观、图表丰富且兼容绝大多数浏览器,能够很简单便捷地在 Web 网站或 Web 应用中添加交互性的图表。Highcharts 目前支持直线图、曲线图、面积图、柱状图、饼图、散点图等多达 18 种不同类型的图表,可以满足你对 Web 图表的任何需求。

Highcharts 的图表结构与名词解释

Highcharts 的图表结构如图 5-66 所示。

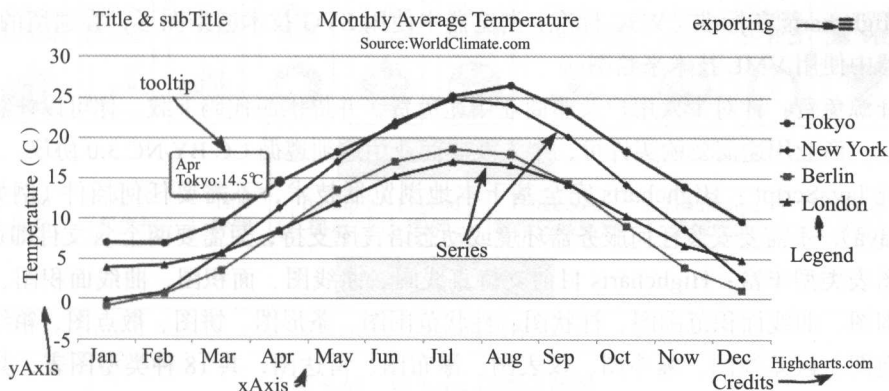


图 5-66 Hightcharts 的图表结构

Highcharts 各部分内容描述如表 5-9 所示。

表 5-9 HighCharts 的图表结构描述

英文名	中文名	描述
lang	语言文字对象	所有 Highcharts 文字相关的设置
chart	图表	图表区、图形区和通用图表配置选项
colors	颜色	图表数据列颜色配置, 是一个颜色数组

(续)

英文名	中文名	描述
credits	版权信息	Highcharts 在图表的右下方放置的版权信息及链接
drilldown	向下钻取	向下钻取数据，深入到其中的具体数据
exporting	导出模块	导出功能配置，导出即将图表下载为图片或打印图表
labels	标签	可以放置到图表区域内任何位置的 HTML 标签
legend	图例	用不同形状、颜色、文字等标示不同数据列，通过点击标示可以显示或隐藏该数据列
loading	加载中	加载选项控制覆盖绘图区的加载屏的外观和文字
navigation	导航	导出模块按钮和菜单配置选项组
noData	没有数据	没有数据时显示的内容
pane	分块	针对仪表盘图和雷达图专用的配置，主要设置弧度及背景色
plotOptions	数据点配置	针对不同类型图表的配置。Highcharts 所有图表类型请看下表
series	数据列	图表上一个或多个数据系列，例如图表中的一条曲线、一个柱形
title	标题	包括标题和副标题，其中副标题为非必须的
tooltip	数据点提示框	当鼠标滑过某点时，以框的形式提示该点的数据，例如该点的值、数据单位等
Axis	坐标轴	包括 x 轴和 y 轴。多个不同的数据列可共用同一个 x 轴或 y 轴，当然，还可以有两个 x 轴或 y 轴，分别显示在图表的上下或左右

Highcharts 的优势

- ❑ 兼容性：Highcharts 支持目前所有的现代浏览器，包括 IE6 +、iPhone/iPad、Android。Highcharts 在标准（W3C 标准）浏览器中使用 SVG 技术渲染图形，在遗留的 IE 浏览器中使用 VML 技术来绘图。
- ❑ 开源免费：针对个人用户及非商业用途免费，并提供源代码下载，你可以任意地修改它。商业用途需要购买许可，个人及非商业用途须遵循 CC BY-NC 3.0 协议。
- ❑ 纯 JavaScript：Highcharts 完全基于本地浏览器技术，不需要任何插件（例如 Flash、Java），不需要安装任何服务器环境或动态语言库支持，只需要两个 js 文件即可运行。
- ❑ 图表类型丰富：Highcharts 目前支持直线图、曲线图、面积图、曲线面积图、面积范围图、曲线面积范围图、柱状图、柱状范围图、条形图、饼图、散点图、箱线图、气泡图、误差线图、漏斗图、仪表图、瀑布图、雷达图，共 18 种类型图表，其中很多图表可以集成在同一个图形中形成综合图。
- ❑ 动态性：提供丰富的 API 接口，方便在创建图表后对图表的任意点、线和文字等进行增加、删除和修改操作。支持众多的 JavaScript 事件，结合 jQuery、MooTools、Prototype 等 JavaScript 框架提供的 Ajax 接口，可以实时地从服务器取得数据并实时刷新图表。
- ❑ 多轴支持：对于需要比较的数据，Highcharts 提供多轴支持，并且可以针对每个轴设置其位置、文字和样式等属性。
- ❑ 动态提示框：当鼠标悬停在图表上的数据点时，Highcharts 会显示信息提示框，当然，

显示的内容和样式可以自己指定和设置。

- ❑ 图表导出和打印功能：你可以将 Highcharts 图表导出为 PNG、JPG、PDF 和 SVG 格式文件或直接在网页上打印出来。
- ❑ 图表缩放：可以设置图表的缩放，让你更方便查看图表数据。
- ❑ 支持外部数据加载：Highcharts 支持多种数据形式，可以是 JavaScript 数组、json 文件、json 对象和表格数据等，这些数据来源可以是本地、不同页面，甚至是不同网站。

4. ECharts

ECharts，纯 JavaScript 图表库，基于 Canvas，底层依赖 ZRender，为商业产品常用图表库，提供直观、生动、可交互、可个性化定制的数据可视化图表。创新的拖拽重计算、数据视图、值域漫游等特性大大增强了用户体验，赋予了用户对数据进行挖掘、整合的能力。图表类型支持折线图（区域图）、柱状图（条状图）、散点图（气泡图）、K 线图、饼图（环形图）、雷达图（填充雷达图）、和弦图、力导布局图、地图、仪表盘以及漏斗图，同时支持任意维度的堆积和多图表混合展现。ECharts 功能示例如图 5-67 所示。

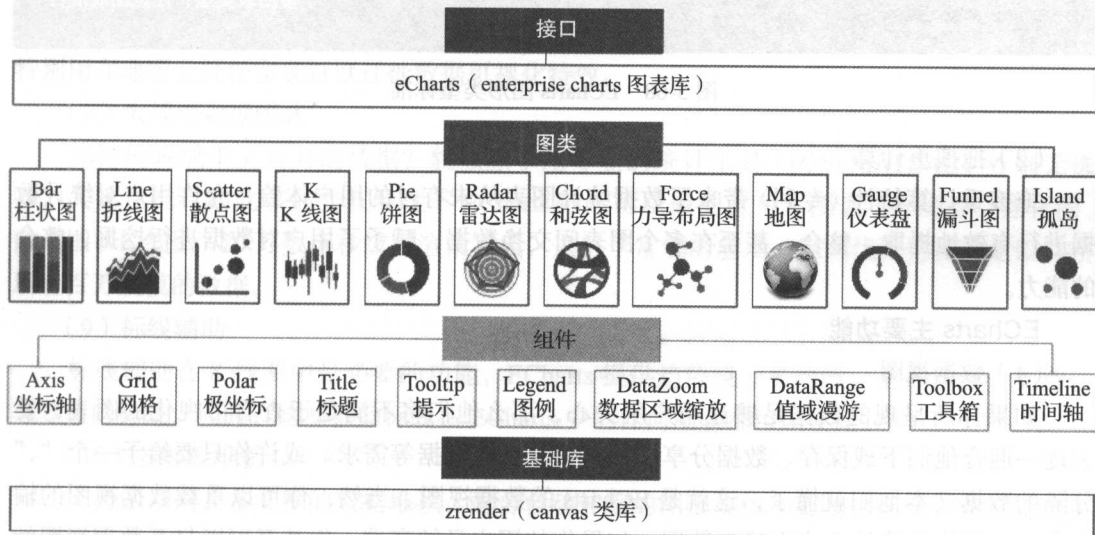


图 5-67 ECharts 功能示例图

ECharts 特色

深度数据互动可视化

打破单纯的视觉呈现，拥有互动图形用户界面（GUI）的数据可视化。数据呈现不仅是诉说，而是允许用户对所呈现的数据进行挖掘、整合，让可视化成为辅助人们进行视觉化思考的方式。

（1）混搭

混搭的图表会更具表现力也更有趣味，ECharts 提供的图表（共 11 类 17 种）支持任意混

搭：折线图（面积图）、柱状图（条形图）、散点图（气泡图）、K线图、饼图（环形图）、雷达图（填充雷达图）、地图、和弦图、力导布局图、仪表盘、漏斗图。混搭情况下，一个标准图表：包含唯一图例、工具箱、数据区域缩放、值域漫游模块，一个直角坐标系（可包含一条或多条类目轴线，一条或多条值轴线，最多上下左右四条），如图 5-68 所示。

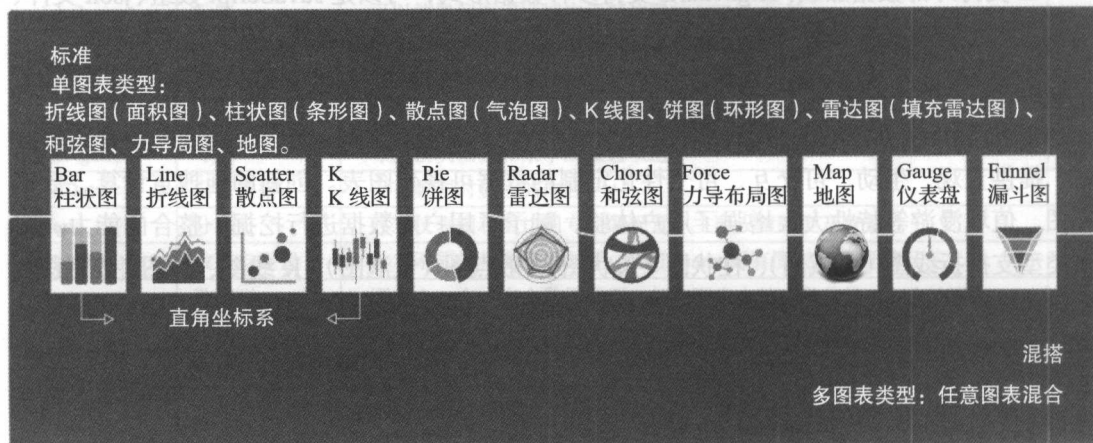


图 5-68 ECharts 图形类型标准

（2）拖拽重计算

拖拽重计算特性（专利）带来了数据统计图表从未有过的用户体验，允许用户对统计数据有效地提取、整合，甚至在多个图表间交换数据，赋予了用户对数据进行挖掘、整合的能力。

ECharts 主要功能

（1）数据视图

如果你所呈现的数据足够让用户所关心，那么他们将不满足于查看可视化的图表，要去逐一迎合他们下载保存、数据分享、加工整合已有数据等需求。或许你只要给予一个“，”分隔的数据文本他们就懂了，这就是 ECharts 的数据视图。当然，你可以重载数据视图的输出方法，用你独特的方式去呈现数据。如果你的用户足够高端，你甚至可以打开数据视图的编辑功能，跟拖拽重计算相比，这可是批量的数据修改。

（2）动态类型切换

很多图表类型本身所表现的能力是相似的，但由于数据差异、表现需求和个人喜好的不同导致最终图表所呈现的张力又大不一样，例如折线图和柱状图的选择，系列数据是堆叠还是平铺总是让人头疼。ECharts 提供了动态类型切换，让用户随心所欲地切换到他所需要的图表类型和堆叠状态。

（3）图例开关

多系列数据的同时展现呈现出丰富的内容，但如何让用户切换到他所关心的个别系列

上, ECharts 提供了方便快捷的多维度图例开关, 可以随时切换到你所关心的数据系列。

(4) 数据区域选择

数据可以是无限的, 但显示空间总是有限的, 数据区域选择组件提供了大数据量中漫游的能力, 让用户选择并呈现他所关心的数据区域。配合随动的均值(极值)标线, 标注展现强大的数据剖析能力。

(5) 多图联动

多系列数据在同一个直角系内同时展现有时候会产生混乱, 但它们又存在极强的关联意义不可分离。ECharts 提供了多图联动的能力(Connect), 能做的可不仅仅是鼠标划过的详情显示, 连接的多个图表会共享组件事件并且实现了保存图片时的自动拼接。

(6) 值域漫游

基于坐标的图表(例如地图、散点图)通过色彩变化表现数值的大小能直观形象地展示数据分布。但如何聚焦到我所关心的数值上? 我们创造了称为值域漫游的功能, 让你可以轻松进行数值筛选。

(7) 炫光特效

我们知道, 很多时候我们需要一些吸引眼球的能力。ECharts 支持标注标线的炫光特效, 特别用在地图上轻松实现百度迁徙数据可视化特效。

(8) 大规模数据模式

如何展现成千上百万的数据? 貌似除了用专业的统计工具(例如 MATLAB) 外别无选择, 其实在拥有如此多的交互特性下 ECharts 依然可以做到直角系图表(折线图、柱状图、散点图、K 线图) 20 万数据秒级内渲染完成, 这对于常规的应用, 用现代浏览器就足以轻松展现百万规模的数据。

(9) 标线辅助

标线辅助在 K 线图中是必要的功能, ECharts 提供趋势线、平均线、上升通道、支持位等专业的标线, 同时, 这些辅助标线在 ECharts 中的任何图表都可以使用。

(10) 多维度堆积

支持多系列、多维度的数据堆积, 配合自动伸缩的图形实体和直角坐标系, 能呈现出更有内涵的统计图表。

(11) 子区域地图模式

地图类型支持 world、china 及全国 34 个省市自治区。同时支持子区域模式, 通过主地图类型扩展出所包含的子区域地图, 轻易输出全球 176 个国家地区和全国 600 多个省市区域简图。

(12) GeoJson 地图扩展

内置地图由标准 GeoJson 地理数据经过高效的压缩算法压缩生成的地图数据(大小仅为标准 GeoJson 的 30% 左右) 驱动而来。如果内置地图类型或数据并未满足你的项目需要, 可通过简单动态注册产生你所需要的新类型。

(13) 标注和标线

添加额外的标注内容是常用的功能，例如地图上标注某些特定位置，折线图上标注极值点或者柱状图上标线出变化趋势。ECharts 全系列图表支持标注标线功能，并且与生俱来地可以响应图例开关、值域漫游等组件的交互功能。

(14) 个性化定制

超过 600 个可配置选项结合多级控制设计满足高度定制的个性化需求。

(15) 事件交互

可以捕获的用户交互和数据变化事件实现图表间或者与外界的联动。

(16) 百搭时间轴

时空数据分析是信息可视化中一个相当重要的方向！ECharts 提供可与任意图表搭配使用的时间轴控件以展现时空数据变化。

ECharts 和 Excel 对比

虽说 Excel 输出的图表毫无交互性可言，但其丰富的图表类型和配置项简单易用，无疑是最常用的制作数据统计的工具。先看看 ECharts 和 Excel 都支持哪些图表类型？如表 5-10 所示。

表 5-10 ECharts 和 Excel 支持图表类型

	ECharts	Excel
柱状图	Yes	Yes
条形图	Yes	Yes
折线图	Yes	Yes
面积图	Yes	Yes
散点图	Yes	Yes
气泡图	Yes	Yes
K 线图	Yes	Yes
饼图	Yes	Yes
环形图	Yes	Yes
雷达图	Yes	Yes
力导布局图	Yes	No
和弦图	Yes	No
曲面图	No	Yes
地图	Yes	No
事件流程图	Yes	No

ECharts 和 Highcharts 对比

业界有无数 js 图表库，不乏优秀的代表，例如 chartjs、FusionCharts、amCharts、flot、RGraph、jqPlot、gRaphaël 等。有的是免费甚至开源的，有的则是商业的，百度一下就能找到它们。无法跟它们进行一一对比，在此选择了知名度很高的 Highcharts，一个优秀、成熟的商业图表

库。先看看 ECharts 和 Highcharts 都支持哪些图表类型和特性，如表 5-11 和表 5-12 所示。

表 5-11 ECharts 和 Highcharts 支持图表类型

	ECharts	Highcharts
柱状图（条形图）	Yes	Yes
折线图（面积图）	Yes	Yes
饼图（环形图）	Yes	Yes
散点图（气泡图）	Yes	Yes
雷达图	Yes	Yes
K 线图	Yes	Highstock
力导布局图	Yes	No
和弦图	Yes	No
地图	Yes	Highmap
事件流程图	Yes	No
特色图表（例如仪表盘）	Yes	Yes

表 5-12 ECharts 和 Highcharts 的特性

	ECharts	Highcharts
拖拽重计算	Yes	No
数据视图	Yes	No
动态类型切换	Yes	No
值域漫游	Yes	No
大规模散点	Yes	No
炫光特效	Yes	No
多图联动	Yes	No
数据区域缩放	Yes	Yes
图例开关	Yes	Yes
多维度堆积	Yes	Yes
混搭	Yes	Yes
图片导出	Yes	Yes
License & Pricing	Free Baidu BSD	Non-commercial free under CC3.0 Commercial licenses \$90~\$3600

我们只是尽我们所能呈现数据真实的一面，并且提供了一些直观、易用的交互方式以方便展现数据进行挖掘、提取、修正或整合（拖拽重计算、数据视图），让大家可以更加专注于所关心的地方，无论是系列选择、区域缩放还是数值筛选（图例开关、数据区域缩放、值域漫游），均可让你有不同的方式解读同样的数据（动态类型切换、多维度堆积、多图联动、混搭）。ECharts 可以完全实现互动图形用户界面（GUI）的数据可视化。

5. D3

D3.js 是一个基于数据的文档操控 JavaScript 库。使用 HTML、SVG 和 CSS，D3 能够帮

你让数据活起来。D3 所强调的 Web 标准帮助开发者在无需捆绑任何专有框架的前提下，结合强大的可视化组件及其数据驱动的 DOM 操纵方法，充分利用现代浏览器的全部功能。如图 5-69 所示为 D3 支持的部分图形。



图 5-69 D3 支持的部分图形

D3 允许开发者将任意数据绑定在文档对象模型 (DOM) 之上，然后再应用数据驱动转换到文档中。比如，你可以使用 D3 从一个数组生成一个 HTML 表格。或者使用同样的数据来创建一个带有平滑过渡和互动功能的交互式 SVG 柱状图。

D3 并非一个旨在涵盖所有功能特征的整体框架，相反，D3 解决的问题核心是：基于数据的高效文档操作。这避免了局限的数据展现，提供了非凡的灵活性，体现出诸如 CSS3、HTML5 和 SVG 等 Web 标准的全部功能。使用最小的开销，D3 的速度非常快，支持大型数据集以及交互与动画的动态行为。D3 的函数风格允许通过各种组件和插件的形式进行代码的重用。

D3 主要功能特点

(1) 选择

使用 W3C DOM API 修改文档是十分繁琐的：方法名冗长，并且必须使用的方法需要人工迭代和对临时状态的维护。比如，要改变段落元素的文本颜色：

```
var paragraphs = document.getElementsByTagName("p");
for (var i = 0; i < paragraphs.length; i++) {
    var paragraph = paragraphs.item(i);
    paragraph.style.setProperty("color", "white", null);
}
```

D3 采用了一种声明式的方法，对任意节点集合的操作被称为选择。比如，你可以这样

重写上面的循环：

```
d3.selectAll("p").style("color", "white");
```

选择器被定义在 W3C 选择器 API 之中并且被现代浏览器原生支持。对于老版本浏览器的向后兼容可以通过 Sizzle 提供。上面的例子中通过标签名称（分别是“p”和“body”）选择节点。元素可以通过使用各种谓词来进行选择，包括容器、属性值、类和 ID。

D3 提供了改变节点的方法：设置属性或者样式；注册事件监听器；对节点进行添加、删除或者排序；以及改变 HTML 或者文本内容。这些可以满足大部分需求。对底层 DOM 的直接访问也是可以的，因为每一个 D3 选择集合只是节点的数组而已。

（2）动态属性

熟悉其他 DOM 框架，例如 jQuery 或者 Prototype 的读者应该可以马上发现它们与 D3 的相似之处。然而在 D3 中，样式、属性和其他一些属性可以通过数据函数的形式指定，而不仅仅限于常量。尽管看起来很简单，但这些函数却是异常强大；比如，d3.geo.path 可以将地理坐标映射在 SVG 路径数据当中。D3 提供了许多内建的可重用函数和函数工厂，例如区域、线条和饼图的图元。

举个例子，随机为段落着色：

```
d3.selectAll("p").style("color", function() {
  return "hsl(" + Math.random() * 360 + ", 100%, 50%)";
});
```

为奇偶节点交替设置灰色阴影：

```
d3.selectAll("p").style("color", function(d, i) {
  return i % 2 ? "#fff" : "#eee";
});
```

计算得出的属性经常指向被绑定的数据。数据是由数值数组指定的，并且每一个数值通过选择函数的第一个参数（d）来传递。使用默认的索引连接（join-by-index），数据数组中的第一个元素传递给选择集合中的第一个节点，第二个元素传给第二个节点，以此类推。比如，如果你将一组数字绑定给段落元素，你可以使用这些数字计算出动态的字体大小：

```
d3.selectAll("p")
  .data([4, 8, 15, 16, 23, 42])
  .style("font-size", function(d) { return d + "px"; });
```

一旦数据与文档绑定，你可以省略数据操作符；D3 会取出先前绑定的数据。这允许你无需重新绑定即可再次对属性进行计算。

（3）输入和退出

使用 D3 的输入选择器和退出选择器可以为新数据创建新节点，移除不再需要的节点。

当数据与选择集合绑定后，数据数组中的每一个元素都与选择集合中的节点一一对应。如果节点比数据少，多余的数据元素形成了输入选择器，你可以通过添加至输入选择器的方式实例化。比如：

```
d3.select("body").selectAll("p").data([4, 8, 15, 16, 23, 42]).enter().append("p").
text(function(d) { return "I'm number " + d + "!"; });
```

更新节点是缺省选择器——数据运算符的结果。因此，如果忘记输入和退出选择器，你会自动地只选择存在对应数据的元素。一种通用的模式是将初始选择集合拆分为 3 部分：要修改的更新节点、要添加的输入节点和要移除的退出节点。

```
// Update...
var p = d3.select("body").selectAll("p")
    .data([4, 8, 15, 16, 23, 42])
    .text(String);
// Enter...
p.enter().append("p")
    .text(String);
// Exit...
p.exit().remove();
```

通过分别处理这 3 种情况，你可以精确地制定某一个节点要执行何种操作。这样可以提高性能，同时可以对过渡提供更好的控制。比如，对于一个条形图，你可能使用旧的比例初始化输入数据条，之后随着数据条的更新和移除将输入数据条过渡为新的比例。

D3 可以让你基于数据来对文档进行过渡；这包括创建和销毁元素。D3 允许你改变一个既存文档来响应用户、随时间改变的动画，或者甚至来自第三方的异步通知。混合方法也是可以的，文档由服务器初始渲染，并且在客户端使用 D3 进行更新。

(4) 转化，而非展示

D3 不是一个新的图形展示库。不同于 Processing、Raphaël 或者 Protovis，标记词汇表直接来源于 Web 标准：HTML、SVG 和 CSS。比如，你可以使用 D3 创建 SVG 元素然后使用外部样式表指定样式。你可以使用组合过滤效果 dashed strokes 和 clipping。如果浏览器厂商在将来引入新的特性，你将能够第一时间使用它们，不需要更新工具包。

最重要的是，D3 很容易使用浏览器内置的元素审查器进行调试：你通过 D3 操作的节点就是浏览器原生支持的节点。

(5) 过渡

D3 对于转化的关注自然地扩展到动画过渡。过渡随着时间逐渐插入样式和属性。渐变可以通过缓动函数进行控制，例如“elastic”“cubic-in-out”和“linear”。D3 的内插器支持基本数据类型，例如数字和嵌入字符串的数字（字体大小、路径数据等），以及符合值。你甚至可以扩展 D3 的差值注册来支持复杂的属性和数据结构。

比如，淡入页面背景为黑色：

```
d3.select("body").transition()
    .style("background-color", "black");
```

或者，使用交错延迟来改变符号地图中的圆圈大小：

```
d3.selectAll("circle").transition()
```



```

    .duration(750)
    .delay(function(d, i) { return i * 10; })
    .attr("r", function(d) { return Math.sqrt(d * scale); });

```

通过只对的确发生了变更的属性进行修改, D3 减少了系统开销并且允许较高帧速率的图形复杂度。D3 还允许经由事件的复杂过渡序列。并且, 你还可以使用 CSS3 过渡; D3 不会替代浏览器的工具箱, 而是以一种更加易用的方式将它暴露出来。

6. Bootstrap

Bootstrap 是由两个 Twitter 员工开发并开源的前端框架。新兴而野心十足的 Bootstrap 是基于 Less 的一套前端工具库, 意图非常明显, 想以一个项目, 整合 Compass、Blueprint、h5bp 的目标功能, 成为 Web 前端的一站式解决方案。

- ❑ 一套完整的基础 CSS 模块, 但不如 Compass 丰富和强大。
- ❑ 一套预定义样式表, 包括一个格子布局系统, 和 Blueprint 提供的差不多, 只是设计风格不一样。
- ❑ 一组基于 JQuery 的 JS 交互插件, 这是 Bootstrap 真正强大的地方, 也是严格意义上可以取代 Blueprint 的原因所在, 这些非常不错的小插件, 包括对话框、下拉导航等, 不但功能完善, 而且十分精致, 正在成为众多 JQuery 项目的默认设计标准。

特别提一下, Bootstrap 使用 [Normalize.css] ([http://necolas.github.com/normalize.css/...](http://necolas.github.com/normalize.css/)) 来进行 Reset CSS, 这一项目已经成为事实标准 (超过 Compass 的 Eric meyer 2.0)。

首先, Bootstrap 出自 Twitter, 大厂出品, 并且开源, 自然久经考验, 减少了测试的工作量。站在巨人的肩膀上, 不重复造轮子。

同时, Bootstrap 的代码有着非常良好的代码规范, 从中也可以学习到很多, 在 Bootstrap 的基础之上创建项目, 日后代码的维护也变得异常简单清晰。

基于 LESS、丰富的 Mixin

Bootstrap 的一大优势就是它是基于 LESS 打造的, 并且也有 SASs 版本。正因为如此, 它一推出就包含了一个非常实用的 Mixin 库任你调用。举个很简单的例子, 当你平时要用到一些 CSS3 属性时, 你要根据不同的浏览器写不同的 -prefix-, 例如圆角 border-radius:

```

-webkit-border-radius: 10px;
-moz-border-radius: 10px;
-border-radius: 10px;

```

但是通过 Bootstrap 给你预设好的 Mixin, 你直接写成这样就可以了:

```
@include border-radius (10px);
```

是不是轻松愉快? 基本常用的 CSS3 Mixin 都帮你整理好了, 你都直接调用就可以了, 在此不一一举例。Bootstrap 是目前最好的基于 Less (Sass) 的前端框架, 丰富而实用的 Mixin 应该是其最好的地方。

可以 Responsive 的栅格系统

Bootstrap 的栅格 (Grid) 系统也很先进, 整个 Grid 系统是可以 Responsive 的! 如果你还不知道 Responsive Design, 那么你太落伍了, 建议你了解之。Bootstrap 已经帮你搭好了实现 Responsive Design 的基础框架, 并且非常容易修改。如果你是一个新手, Bootstrap 可以帮助你在非常短的时间内上手 Responsive Design。

丰富的组件和插件

Bootstrap 的 HTML 组件和 JS 组件非常丰富, 并且代码简洁, 非常易于修改, 你完全可以在其基础之上修改成自己想要的任何样子。这是工作效率的极大提升。

另外, 由于 Bootstrap 的火爆, 又出现了不少围绕 Bootstrap 而开发的插件。其中, 最实用的莫过于 Font Awesome 了。它是一套 icon font, 提供了丰富的 icon 给你选择, 新的 2.0 版又根据网友的意见增加了 70 个新的 icon。

目前, 在一股由 Apple 带领的 Retina 潮流下, 对图片的视网膜屏下的解决方案已经变得越来越有必要了, 而对于 icon, icon font 是完美的解决方案, 你不用担心分辨率的问题, 因为它实际上是字体。

5.2.2 数据缓存

数据缓存 (Data Caching) 就是将数据暂存于内存缓存区中的一种技术, 主要作用是减少数据存储的访问次数与提高数据访问的效率。

数据缓存能解决哪些问题:

- ❑ 性能——将相应数据存储起来以避免数据的重复创建、处理和传输, 可有效提高性能。比如, 将不改变的数据缓存起来, 例如国家列表等, 这样能明显提高 Web 程序的反应速度。
- ❑ 稳定性——同一个应用中, 对同一数据、逻辑功能和用户界面的多次请求是经常发生的。当用户基数很大时, 如果每次请求都进行处理, 消耗的资源是很大的浪费, 同时也造成系统的不稳定。比如, Web 应用中, 对一些静态页面的呈现内容进行缓存能有效地节省资源, 提高稳定性。而缓存数据也能降低对数据库的访问次数, 降低数据库的负担和提高数据库的服务能力。
- ❑ 可用性——有时, 提供数据信息的服务可能会意外停止, 如果使用了缓存技术, 可以在一定时间内仍正常提供对最终用户的支持, 提高了系统的可用性。

1. Memcache

Memcache 介绍

Memcache 是一套分布式的高速缓存系统, 由 LiveJournal 的 Brad Fitzpatrick 开发, 这是一套开放源代码软件, 以 BSD license 授权发布。目前被许多网站使用以提升网站的访问速度。许多 Web 应用都将数据保存到 RDBMS 中, 应用服务器从中读取数据并在浏览器中显

示。但随着数据量的增大、访问的集中,就会出现 RDBMS 的负担加重、数据库响应恶化、网站显示延迟等重大影响。这时 Memcache 就大显身手了,一般的使用目的是,通过缓存数据库的查询结果,减少数据库的访问次数,以提高动态 Web 应用的速度,提高可扩展性。尤其对于一些大型的、需要频繁访问数据库的网站访问速度提升效果十分显著。图 5-70 为 Memcache 的一般用途。

Memcache 的特征

□ 协议简单。它是基于文本行的协议,直接通过 telnet 在 Memcached 服务器上可进行存取数据操作。

□ 基于 libevent 事件处理。libevent 是一套利用 C 开发的程序库,它将 BSD 系统的 kqueue、Linux 系统的 epoll 等事件处理功能封装成一个接口,与传统的 select 相比,提高了性能。

□ 内置的内存管理方式。所有数据都保存在内存中,存取数据比硬盘快,当内存满后,通过 LRU 算法

自动删除不使用的缓存,但没有考虑数据的容灾问题,重启服务,所有数据会丢失。

□ 分布式。各个 Memcached 服务器之间互不通信,各自独立存取数据,不共享任何信息。服务器并不具有分布式功能,分布式部署取决于 Memcache 客户端。

Memcached 的内存算法

Memcached 利用 Slab Allocation 机制来分配和管理内存。在该机制出现以前,内存的分配是通过对所有记录简单地进行 malloc 和 free 来实现的。但是,这种方式会导致内存碎片,加重操作系统内存管理器的负担,最坏的情况下,会导致操作系统比 Memcache 进程本身还慢,Slab Allocation 就是为了解决该问题而诞生的。

先来看看 Slab Allocator 的原理。下面是 Memcached 文档中的 Slab Allocator 的目标:

The primary goal of the slabs subsystem in memcached was to eliminate memory fragmentation issues totally by using fixedsize memory chunks coming from a few predetermined size classes.

也就是说,Slab Allocator 的基本原理是按照预先规定的大小,将分配的内存分割成特定长度的块,以完全解决内存碎片问题。

Slab Allocation 的原理相当简单。将分配的内存分割成各种尺寸的块 (Chunk),并把尺寸相同的块分成组 (Chunk 的集合)。图 5-71 为 Slab Allocation 的构造图。

而且,Slab Allocator 还有重复使用已分配内存的目的。也就是说,分配到的内存不会释

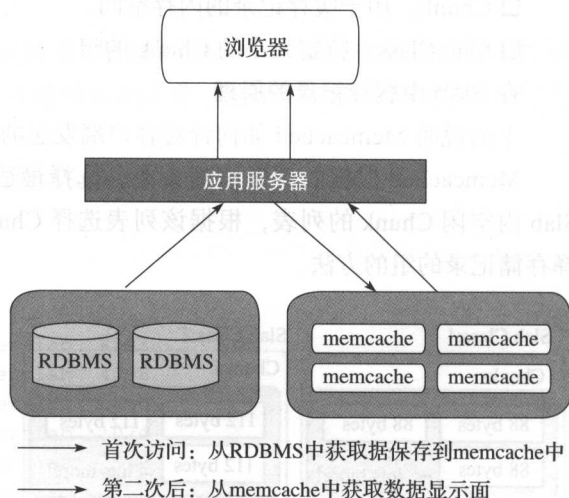


图 5-70 Memcache 的一般用途

放，而是重复利用。

Slab Allocation 的主要术语

- ❑ Page。分配给 Slab 的内存空间，默认是 1MB。分配给 Slab 之后根据 Slab 的大小切分成 Chunk。
- ❑ Chunk。用于缓存记录的内存空间。
- ❑ Slab Class。特定大小的 Chunk 的组。

在 Slab 中缓存记录的原理

下面说明 Memcached 如何针对客户端发送的数据选择 Slab 并缓存到 Chunk 中。

Memcached 根据收到数据的大小，选择最适合数据大小的 Slab。Memcached 中保存着 Slab 内空闲 Chunk 的列表，根据该列表选择 Chunk，然后将数据缓存于其中。图 5-72 为选择存储记录的组的方法。

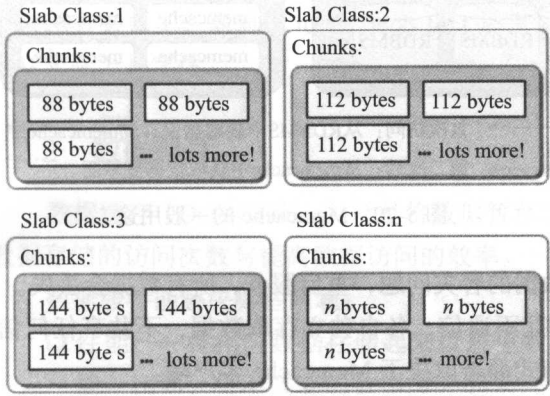


图 5-71 Slab Allocation 的构造图

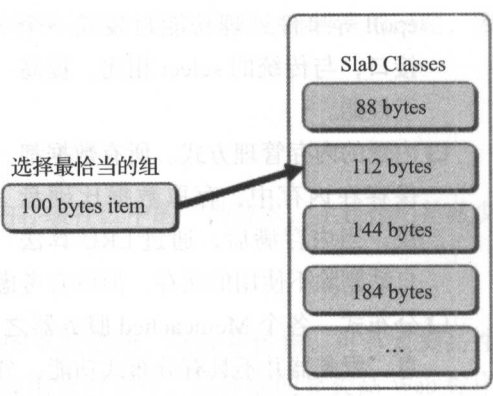


图 5-72 选择存储记录的组的方法

Slab Allocator 的缺点

Slab Allocator 解决了当初的内存碎片问题，但新的机制也给 Memcached 带来了新的问题。

这个问题就是，由于分配的是特定长度的内存，因此无法有效利用分配的内存。比如，将 100 字节的数据缓存到 128 字节的 Chunk 中，剩余的 28 字节就浪费了。图 5-73 为 Chunk 空间的使用示意图。

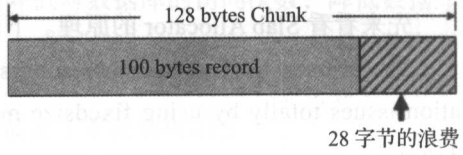


图 5-73 Chunk 空间的使用

对于该问题目前还没有完美的解决方案，但在文档中记载了比较有效的解决方案。

The most efficient way to reduce the waste is to use a list of size classes that closely matches (if that's at all possible) common sizes of objects that the clients of this particular installation of memcached are likely to store.

就是说,如果预先知道客户端发送数据的公用大小,或者仅缓存大小相同的数据的情况下,只要使用适合数据大小的组的列表,就可以减少浪费。

但是很遗憾,现在还不能进行任何调优,只能期待以后的版本了。但是,我们可以调节 Slab Class 的大小的差别。

使用 Growth Factor 进行 Slab 调优

Memcached 在启动时指定 Growth Factor 因子(通过 f 选项),就可以在某种程度上控制 Slab 之间的差异,默认值为 1.25。但是,在该选项出现之前,这个因子曾经固定为 2,称为“powers of 2”策略。

让我们用以前的设置,以 verbose 模式启动 Memcached 试试看:

```
$ memcached f 2 vv
```

下面是启动后的 verbose 输出:

```
slab class 1: chunk size 128 perslab 8192
slab class 2: chunk size 256 perslab 4096
slab class 3: chunk size 512 perslab 2048
slab class 4: chunk size 1024 perslab 1024
slab class 5: chunk size 2048 perslab 512
slab class 6: chunk size 4096 perslab 256
slab class 7: chunk size 8192 perslab 128
slab class 8: chunk size 16384 perslab 64
slab class 9: chunk size 32768 perslab 32
slab class 10: chunk size 65536 perslab 16
slab class 11: chunk size 131072 perslab 8
slab class 12: chunk size 262144 perslab 4
slab class 13: chunk size 524288 perslab 2
```

可见,从 128 字节的组开始,组的大小依次增大为原来的 2 倍。这样设置的问题是,Slab 之间的差别比较大,有些情况下就相当浪费内存。因此,为尽量减少内存浪费,追加了 Growth Factor 这个选项。

来看看现在的默认设置(f=1.25)时的输出:

```
slab class 1: chunk size 88 perslab 11915
slab class 2: chunk size 112 perslab 9362
slab class 3: chunk size 144 perslab 7281
slab class 4: chunk size 184 perslab 5698
slab class 5: chunk size 232 perslab 4519
slab class 6: chunk size 296 perslab 3542
slab class 7: chunk size 376 perslab 2788
slab class 8: chunk size 472 perslab 2221
slab class 9: chunk size 592 perslab 1771
slab class 10: chunk size 744 perslab 1409
```

可见,组间差距比因子为 2 时小得多,更适合缓存几百字节的记录。从上面的输出结果来看,你可能会觉得有些计算误差,但这些误差是为了保持字节数的对齐而故意设置的。

将 Memcached 引入产品，或是直接使用默认值进行部署时，最好是重新计算一下数据的预期平均长度，调整 Growth Factor，以获得最恰当的设置。内存是珍贵的资源，浪费就太可惜了。

查看 Slabs 的使用状况

使用 Memcached 创造着 Brad 写的名为 memcachedtool 的 Perl 脚本，可以方便地获得 Slab 的使用情况（它将 Memcached 的返回值整理成容易阅读的格式）。可以从下面的地址获得脚本：<http://code.sixapart.com/svn/memcached/trunk/server/scripts/memcachedtool>。

使用方法也极其简单：

```
$ memcachedtool 主机名: 端口选项
```

查看 Slabs 使用状况时无需指定选项，因此用下面的命令即可：

```
$ memcachedtool 主机名: 端口
```

获得的信息如下所示：

#	Item_Size	Max_age	1MB_pages	Count	Full?
1	104 B	1394292 s	1215	12249628	yes
2	136 B	1456795 s	52	400919	yes
3	176 B	1339587 s	33	196567	yes
4	224 B	1360926 s	109	510221	yes
5	280 B	1570071 s	49	183452	yes
6	352 B	1592051 s	77	229197	yes
7	440 B	1517732 s	66	157183	yes
8	552 B	1460821 s	62	117697	yes
9	696 B	1521917 s	143	215308	yes
10	872 B	1695035 s	205	246162	yes
11	1.1 kB	1681650 s	233	221968	yes
12	1.3 kB	1603363 s	241	183621	yes
13	1.7 kB	1634218 s	94	57197	yes
14	2.1 kB	1695038 s	75	36488	yes
15	2.6 kB	1747075 s	65	25203	yes
16	3.3 kB	1760661 s	78	24167	yes

各列的含义如表 5-13 所示。

表 5-13 信息中各列的含义

列	含义
#	Slab Class 编号
Item_Size	Chunk 大小
Max_age	LRU 内最旧的记录的生存时间
1MB_pages	分配给 Slab 的页数
Count	Slab 内的记录数
Full ?	Slab 内是否含有空闲 Chunk

Memcached 的缓存策略

Memcached 的缓存策略是 LRU (Least Recently Used, 最近最少使用) 加上到期失效策略。当你在 Memcached 内存存储数据项时, 你有可能会指定它在缓存的失效时间, 默认为永久。当 Memcached 服务器用完分配的内存时, 失效的数据被首先替换, 然后就是最近未使用的数据。在 LRU 中, Memcached 使用的是一种 Lazy Expiration 策略, 自己不会监控存入的 key/value 对是否过期, 而是在获取 key 值时查看记录的时间戳, 检查 key/value 对空间是否过期, 这样可减轻服务器的负载。

Memcache 的分布式策略

当向 Memcached 集群存入 / 取出 key/value 时, Memcached 客户端程序根据一定的算法计算存入哪台服务器, 然后再把 key/value 值存到此服务器中。也就是说, 存取数据分两步走, 第一步, 选择服务器, 第二步存取数据。图 5-74 为 Memcache 分布式策略示意图。

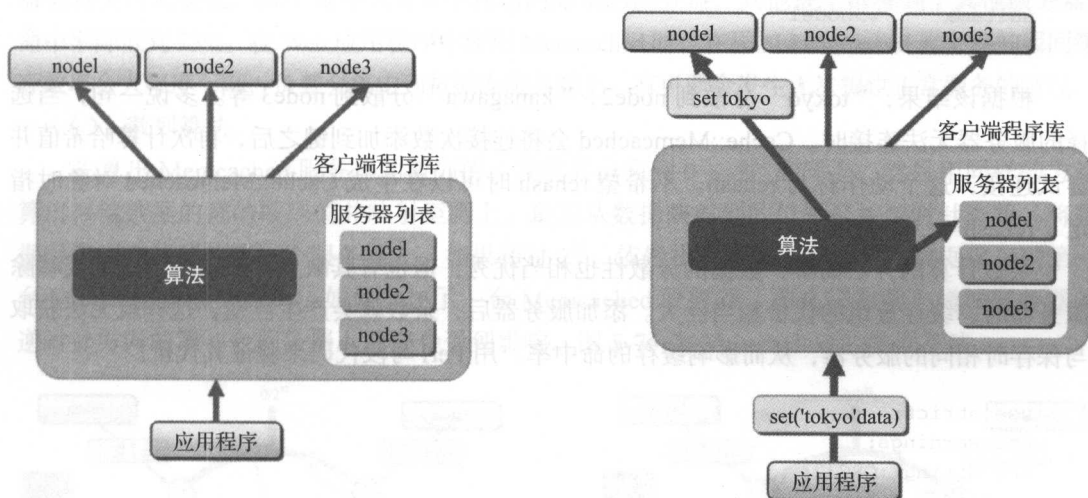


图 5-74 memcache 分布式策略

Memcache 计算分布的算法

选择服务器算法有两种, 一种是根据余数来计算分布, 另一种是根据散列算法来计算分布。

(1) 余数算法

利用 Memcache 的余数来计算分布的方法简单来说, 就是“根据服务器台数的余数进行分散”。先求得键的整数哈希值, 再除以服务器台数, 根据其余数来选择服务器。

下面将 Cache::Memcached 简化成以下的 Perl 脚本来进行说明。

```
use strict;
use warnings;
use String::CRC32;
my @nodes = ('node1', 'node2', 'node3');
my @keys = ('tokyo', 'kanagawa', 'chiba', 'saitama', 'gunma');
foreach my $key (@keys) {
```

```
my $src = crc32 ($key); # CRC 值
my $mod = $src % ( $#nodes + 1 );
my $server = $nodes[ $mod ]; # 根据余数选择服务器
printf "%s => %s\n", $key, $server;
}
```

Cache::Memcached 在求哈希值时使用了 CRC。

- String::CRC32 search.cpan.org

首先求得字符串的 CRC 值，根据该值除以服务器节点数目得到的余数决定服务器。上面的代码执行后输入以下结果：

```
tokyo      =>node2
knagawa    =>node3
chiba      =>node2
saitama    =>node1
gunma      =>node1
```

根据该结果，“tokyo”分散到 node2，“kanagawa”分散到 node3 等。多说一句，当选择的服务器无法连接时，Cache::Memcached 会将连接次数添加到键之后，再次计算哈希值并尝试连接，这个动作称为 rehash。不希望 rehash 时可以在生成 Cache::Memcached 对象时指定“rehash=>0”选项。

余数计算的方法简单，数据的分散性也相当优秀，但也有其缺点。那就是当添加或删除服务器时，缓存重组的代价相当巨大。添加服务器后，余数就会产生巨变，这样就无法获取与保存时相同的服务器，从而影响缓存的命中率。用 Perl 写段代码来验证其代价。

```
use strict;
use warnings;
use String::CRC32;
my @nodes = @ARGV;
my @keys = ('a'..'z');
my %nodes;
foreach my $key ( @keys ) {
    my $hash = crc32 ($key);
    my $mod = $hash % ( $#nodes + 1 );
    my $server = $nodes[ $mod ];
    push @{ $nodes{ $server } }, $key;
}
foreach my $node ( sort keys %nodes ) {
    printf "%s: %s\n", $node, join ", ", @{ $nodes{ $node } };
}
```

这段 Perl 脚本演示了将“a”到“z”的键保存到 Memcached 并访问的情况。将其保存为 mod.pl 并执行。首先，当服务器只有 3 台时：

```
$ mod.pl node1 node2 nod3
```

```
node1: a, c, d, e, h, j, n, u, w, x
node2: g, i, k, l, p, r, s, y
node3: b, f, m, o, q, t, v, z
```

结果如上, node1 保存 a、c、d、e……, node2 保存 g、i、k……, 每台服务器都保存了 8 ~ 10 个数据。

接下来增加一台 Memcached 服务器。

```
$ mod.pl node1 node2 node3 node4
node1: d, f, m, o, t, v
node2: b, i, k, p, r, y
node3: e, g, l, n, u, w
node4: a, c, h, j, q, s, x, z
```

添加了 node4。可见, 只有 d、i、k、p、r、y 命中了。像这样, 添加节点后键分散到的服务器, 会发生巨大变化。26 个键中只有 6 个在访问原来的服务器, 其他键全都移到了其他服务器, 命中率降低到 23%。在 Web 应用程序中使用 Memcached 时, 在添加 Memcached 服务器的瞬间缓存效率会大幅度下降, 负载会集中到数据库服务器上, 有可能会发生无法提供正常服务的情况。

(2) 散列算法

先算出 Memcached 服务器的散列值, 并将其分布到 $0 \sim 2^{32}$ 的圆上, 然后用同样的方法算出存储数据的键的散列值并映射至圆上, 最后从数据映射到的位置开始顺时针查找, 将数据保存到查找到的第一个服务器上, 如果超过 2^{32} , 依然找不到服务器, 就将数据保存到第一台 Memcached 服务器上。如果添加了一台 Memcached 服务器, 那么只在圆上增加服务器的逆时针方向的第一台服务器上的键会受到影响。图 5-75 为 Memcached 的散列算法。

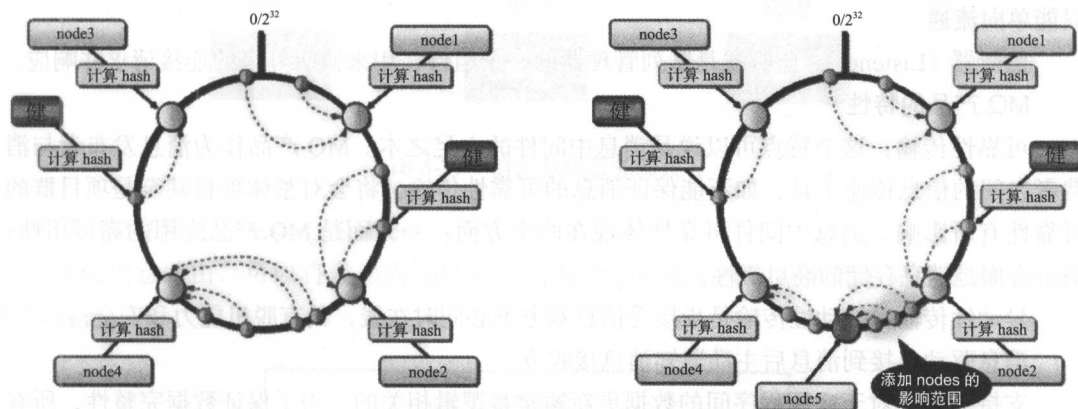


图 5-75 Memcached 的散列算法

5.2.3 中间件

消息中间件, 也可叫作 MOM (Message Oriented Middleware), 是一种由消息传送机制和消息队列模式组成的中间件技术, 利用高效可靠的消息传递机制进行平台无关的数据交

流，并基于数据通信来进行分布式系统的集成。

采用消息中间件机制的系统中，不同的对象之间通过传递消息来激活对方的事件，完成相应的操作。发送者将消息发送给消息中间件，消息中间件将消息存放在若干队列中，在合适的时候再将消息转发给接收者。

消息中间件能在不同平台之间通信，它常被用来屏蔽掉各种平台及协议之间的特性，实现应用程序之间的协同，其优点在于能够在客户和服务器之间提供同步和异步的连接，并且在任何时刻都可以将消息进行传送或者存储转发，这也是它比远程过程调用更进一步的原因。

消息中间件主要有三种工作模式：

- ☐ 点对点模式。
- ☐ 发布 / 订阅模式。
- ☐ 消息队列模式。

1. MQ 系列

MQ 相关概念

消息 (Message)：消息是 MQ 进行数据交换的基本信息单位，本质就是一段数据，是应用程序之间传递的信息载体。

队列 (Queue)：队列是一种用于存储消息的数据结构，可以简单把队列看成一个消息的容器。每个队列在有用该队列的队列管理器中必须用一个唯一命名的名字。

队列管理器 (Queue Manager)：队列管理器是一个负责向应用程序通过消息服务的机构，用来维护和管理消息队列。

通道 (Channel)：通道是两个管理器之间的一种单向点对点的通信连接，消息在通道中只能单向流通。

监听器 (Listener)：监听器是队列管理器的一个组件，用来监听外来的连接请求并响应。

MQ 产品的特性

可靠性传输：这个特点可以说是消息中间件的立足之本。MQ 产品作为信息发布者与消费者之间的信息传递工具，如不能保证消息的可靠性传输，将会对整体项目甚至是项目群的可靠性有所影响。消息中间件可靠性体现在两个方面：一方面是 MQ 产品应用的高可用性；另一方面是消息存储的高可靠性。

异步性传输：异步性传输是指接受信息双方不必同时在线，具有脱机能力和安全性。

消息驱动：接到消息后主动通知消息接收方。

支持事务：对于应用程序间的数据更新通常是逻辑相关的，为了保证数据完整性，所有的更新必须同时成功或者同时失败，所以好的 MQ 产品需有支持事务的特性。

MQ 适用场景介绍

MQ 消息队列是应运松耦合的概念而产生的，主要以队列和发布订阅为消息传输机制，以异步的方式将消息可靠地传输到消费段的一种基础产品。它被广泛地应用于跨平台、跨系统的分布式系统之间，为它们提供高效可靠的异步传输机制。MQ 的应用主要适用于以下几个场景：

(1) 消息通道 (Message Channel)

使用 MQ 将彼此协作的客户端和服务端连接起来,使它们可以交换消息,如图 5-76 所示。

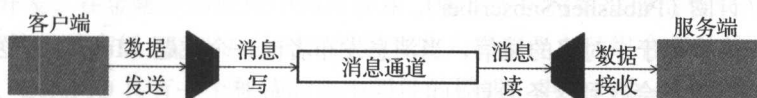


图 5-76 信息通道工作流程

如客户端与服务端需要安全可靠的交互,可以将一个 MQ 的队列作为安全通道,使客户端与服务端能够安全高效地进行异步通信。

(2) 消息总线 (Message Bus)

对于由许多独立开发的服务组成的分布式系统,倘若要将它们组成一个完整的系统,这些服务必须能够可靠地交互,同时,为了系统的健壮性,每个服务之间又不能产生过分紧密的依赖关系,这样就可以通过消息总线将不同的服务连接起来,允许它们异步地传递数据(如图 5-77 所示)。

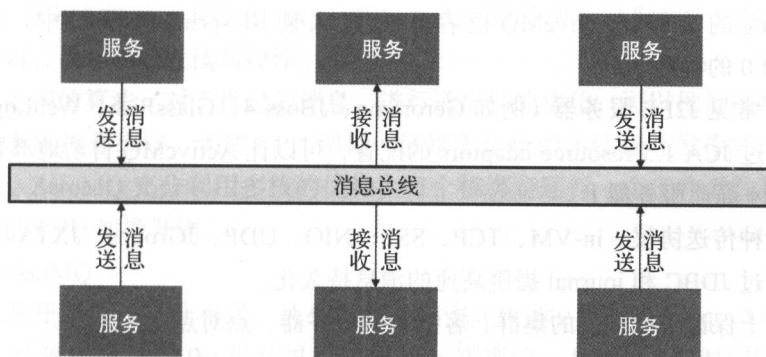


图 5-77 消息总线服务模式

(3) 消息路由 (Message Router)

通过消息路由,可以将发送到 MQ 指定队列的消息根据规则路由到不同的队列,如图 5-78 所示。

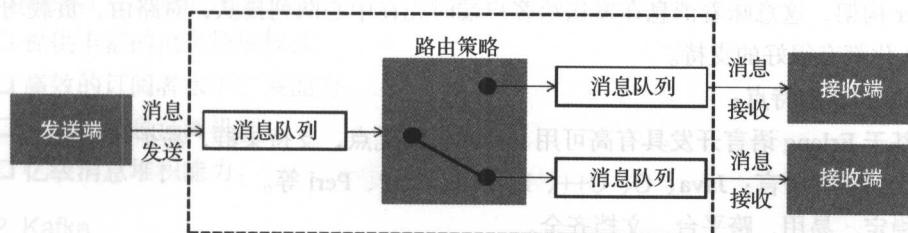


图 5-78 消息路由工作模式

此外，JMS 规范还支持通过 selector 条件，对消息进行过滤，可以用多个消费者消费同一个队列的消息，每个消费者只消费自己感兴趣的消息。

(4) 发布 / 订阅 (Publisher/Subscriber)

发布 / 订阅模式用于一对多的通信，当消息发布者向一个主题 (Topic) 发送一条消息后，该主题的所有订阅者都会收到这条消息。

常用 MQ 产品介绍

(1) ActiveMQ

ActiveMQ 是 Apache 软件基金下的一个开源软件，是目前能力强劲、应用广泛的开源消息总线之一。它为企业消息传递提供高可用、出色性能、可扩展、稳定和安全保障。ActiveMQ 的目标是在尽可能多的平台和语言上提供一个标准的、消息驱动的应用集成。

ActiveMQ 的特点

- ❑ 多种语言的支持：Java、C、C++、Python、PHP、Perl、.net 等。
- ❑ 多种协议的支持：OpenWire、STOMP、REST、XMPP、AMQP。
- ❑ 完全支持 JMS1.1 和 J2EE 1.4 规范。
- ❑ 对 Spring 的支持，ActiveMQ 很容易内嵌入使用 Spring 的项目中，而且也支持 Spring2.0 的特性。
- ❑ 通过了常见 J2EE 服务器（例如 Geronimo、JBoss 4、GlassFish、WebLogic）的测试，其中通过 JCA 1.5 resource adaptors 的配置，可以让 ActiveMQ 自动地部署到任何兼容 J2EE 1.4 商业服务器上。
- ❑ 支持多种传送协议：in-VM、TCP、SSL、NIO、UDP、JGroups、JXTA。
- ❑ 支持通过 JDBC 和 journal 提供高速的消息持久化。
- ❑ 从设计上保证了高性能的集群、客户端 - 服务器、点对点。
- ❑ 支持 Ajax。
- ❑ 支持与 Axis 的整合。
- ❑ 可以很容易地调用内嵌 JMS provider 进行测试。

(2) RabbitMQ

RabbitMQ 是使用 Erlang 编写的一个开源的消息队列，本身支持很多协议：AMQP、XMPP、SMTP、STOMP，也正因如此，它非常重量级，更适合于企业级的开发。同时实现了 Broker 构架，这意味着消息在发送给客户端时先在中心队列排队，对路由、负载均衡或者数据持久化都有很好的支持。

RabbitMQ 的特点：

- ❑ 基于 Erlang 语言开发具有高可用、高并发的优点，支持集群，易扩展。
- ❑ 支持多种语言：Java、C、C++、Python、PHP、Perl 等。
- ❑ 稳定、易用、跨平台、文档齐全。
- ❑ 强大的管理功能。

(3) ZeroMQ

ZeroMQ 又称 ØMQ、0MQ、ZMQ，号称最快的消息队列系统，专门为高吞吐量/低延迟的场景开发，在金融界的应用中经常使用，偏重于实时数据通信场景。ZMQ 能够实现 RabbitMQ 不擅长的高级/复杂的队列，但是开发人员需要自己组合多种技术框架，开发成本高。因此，ZeroMQ 具有一个独特的非中间件的模式，更像一个 socket library，你不需要安装和运行一个消息服务器或中间件，因为你的应用程序本身就是使用 ZeroMQ API 完成逻辑服务的角色。但是 ZeroMQ 仅提供非持久性的队列，如果 down 机，数据将会丢失。比如，Twitter 的 Storm 中使用 ZeroMQ 作为数据流的传输。

ZeroMQ 套接字是与传输层无关的：ZeroMQ 套接字对所有传输层协议定义了统一的 API 接口。默认支持进程内 (inproc)、进程间 (IPC)、多播、TCP 协议，在不同的协议之间切换只要简单地改变连接字符串的前缀。可以在任何时候以最小的代价从进程间的本地通信切换到分布式下的 TCP 通信。ZeroMQ 在背后处理连接建立，断开和重连逻辑。

ZeroMQ 的特点：

- 无锁的队列模式：对于跨线程间的交互（用户端和 session）之间的数据交换通道 pipe，采用无锁的队列算法 CAS；在 pipe 的两端注册有异步事件，在读或者写消息到 pipe 时，会自动触发读写操作。
- 批量处理的算法：对于批量的消息，进行适应性的优化，可以批量接收和发送信息。
- 多核下的线程绑定，无须 CPU 切换：区别于传统的多线程并发模式、信号量或者临界区，ZeroMQ 充分利用多核的优势，每个核绑定运行一个工作者线程，避免多线程之间的 CPU 切换开销。

(4) RocketMQ

阿里系下开源的一款分布式、队列模型的消息中间件，原名 Metaq，3.0 版本名称改为 RocketMQ，是阿里参照 Kafka 设计思想使用 Java 实现的一套 MQ。同时将阿里系内部多款 MQ 产品 (Notify、metaq) 进行整合，只维护核心功能，去除了所有其他运行时依赖，保证核心功能最简化，在此基础上配合阿里上述其他开源产品实现不同场景下 MQ 的架构，目前主要多用于订单交易系统。

RocketMQ 的特点：

- 能够保证严格的消息顺序。
- 提供针对消息的过滤功能。
- 提供丰富的消息拉取模式。
- 高效的订阅者水平扩展能力。
- 实时的消息订阅机制。
- 亿级消息堆积能力。

2. Kafka

Kafka 是 Apache 下的分布式发布-订阅的开源消息系统。它最初是由 LinkedIn 公司开

发，之后成为 Apache 项目的一部分。Kafka 是一种快速、可扩展的、设计内在就是分布式的、分区的和可复制的提交日志服务。

Kafka 是一种高吞吐量的分布式发布订阅消息系统，它可以处理消费者规模的网站中的所有动作流数据。这种动作（网页浏览、搜索和其他用户的行动）是在现代网络上的许多社会功能的一个关键因素。这些数据通常是由于吞吐量的要求而通过处理日志和日志聚合来解决。对于像 Hadoop 一样的日志数据和离线分析系统，但又有时有实时处理限制的要求，这是一个可行的解决方案。Kafka 的目的在于通过 Hadoop 的并行加载机制来统一线上和离线的消息处理，也是为了通过集群机来提供实时的消费。Kafka 架构如图 5-79 所示。

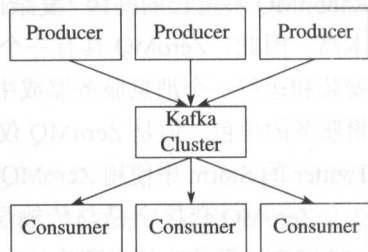


图 5-79 Kafka 架构图

Kafka 是一种高吞吐量的分布式发布订阅消息系统，有如下特性：

- ❑ 通过 O (1) 的磁盘数据结构提供消息的持久化，这种结构对于即使数以 TB 的消息存储也能够保持长时间的稳定性能。
- ❑ 高吞吐量：即使是非常普通的硬件 Kafka 也可以支持每秒数十万的消息。
- ❑ 支持通过 Kafka 服务器和消费机集群来分区消息。
- ❑ 支持 Hadoop 并行数据加载。

Kafka 相关术语：

- ❑ Broker：Kafka 集群包含一个或多个服务器，这种服务器被称为 Broker。
- ❑ Topic：每条发布到 Kafka 集群的消息都有一个类别，这个类别被称为 Topic（物理上不同 Topic 的消息分开存储，逻辑上一个 Topic 的消息虽然保存于一个或多个 Broker 上，但用户只需指定消息的 Topic 即可生产或消费数据而不必关心数据存于何处）。
- ❑ Partition：Partition 是物理上的概念，每个 Topic 包含一个或多个 Partition。
- ❑ Producer：负责发布消息到 Kafka Broker。
- ❑ Consumer：消息消费者，向 Kafka Broker 读取消息的客户端。
- ❑ Consumer Group：每个 Consumer 属于一个特定的 Consumer Group（可为每个 Consumer 指定 group name，若不指定 group name 则属于默认的 group）。

Kafka 拓扑结构：

如图 5-80 所示，一个典型的 Kafka 集群的拓扑结构图，包含若干个 Producer，若干个 Broker（Kafka 支持水平扩展，一般 Broker 的数据越多，集群的吞吐率越高），若干个 Consumer Group，以及一个 Zookeeper 集群。Kafka 通过 Zookeeper 管理集群配置，选举 Leader，以及在 Consumer Group 发生变化时进行 rebalance。Producer 使用 push 模式将消息发布到 Broker，Consumer 使用 pull 模式从 broker 订阅并消费消息。

Topic&Partition

Topic 在逻辑上可以被认为是一个 queue，每条消费都必须指定它的 Topic，可以简单理

解为必须指明把这条消息放进哪个 queue 里。为了使 Kafka 的吞吐率可以线性提高，物理上把 Topic 分成一个或多个 Partition，每个 Partition 在物理上对应一个文件夹，该文件夹下存储这个 Partition 的所有消息和索引文件。

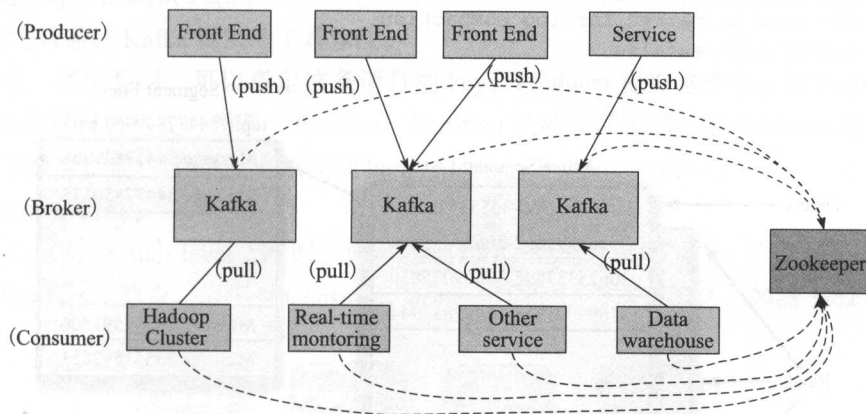


图 5-80 Kafka 集群的拓扑结构图

每个日志文件都是一个 log entry 序列，每个 log entry 包含一个 4 字节整型数值（值为 $N+5$ ），1 个字节的 “magic value”，4 个字节的 CRC 校验码，其后跟 N 个字节的消息体。每条消息都有一个当前 Partition 下唯一的 64 字节的 offset，它指明了这条消息的起始位置。

```
message length : 4 bytes (value: 1+4+n)
"magic" value : 1 byte
crc : 4 bytes
payload : n bytes
```

这个 log entries 并非由一个文件构成，而是分成多个 segment，每个 segment 以该 segment 第一条消息的 offset 命名并以 “.kafka” 为后缀。另外会有一个索引文件，它标明了每个 segment 下包含的 log entry 的 offset 范围，如图 5-81 所示。

因为每条消息都被 append 到该 Partition 中，属于顺序写磁盘，因此效率非常高（经验证，顺序写磁盘效率比随机写内存还要高，这是 Kafka 高吞吐率的一个很重要的保证）。

对于传统的 message queue 而言，一般会删除已经被消费的消息，而 Kafka 集群会保留所有的消息，无论其被消费与否。当然，因为磁盘限制，不可能永久保留所有数据（实际上也没必要），因此 Kafka 提供两种策略删除旧数据，一是基于时间，二是基于 Partition 文件大小。比如，可以通过配置 `$KAFKA_HOME/config/server.properties`，让 Kafka 删除一周前的数据，也可在 Partition 文件超过 1GB 时删除旧数据，配置如下所示。

```
# The minimum age of a log file to be eligible for deletion
log.retention.hours=168
# The maximum size of a log segment file. When this size is reached a new log
segment will be created.
```

```
log.segment.bytes=1073741824
# The interval at which log segments are checked to see if they can be deleted
according to the retention policies
log.retention.check.interval.ms=300000
# If log.cleaner.enable=true is set the cleaner will be enabled and individual
logs can then be marked for log compaction.
log.cleaner.enable=false
```

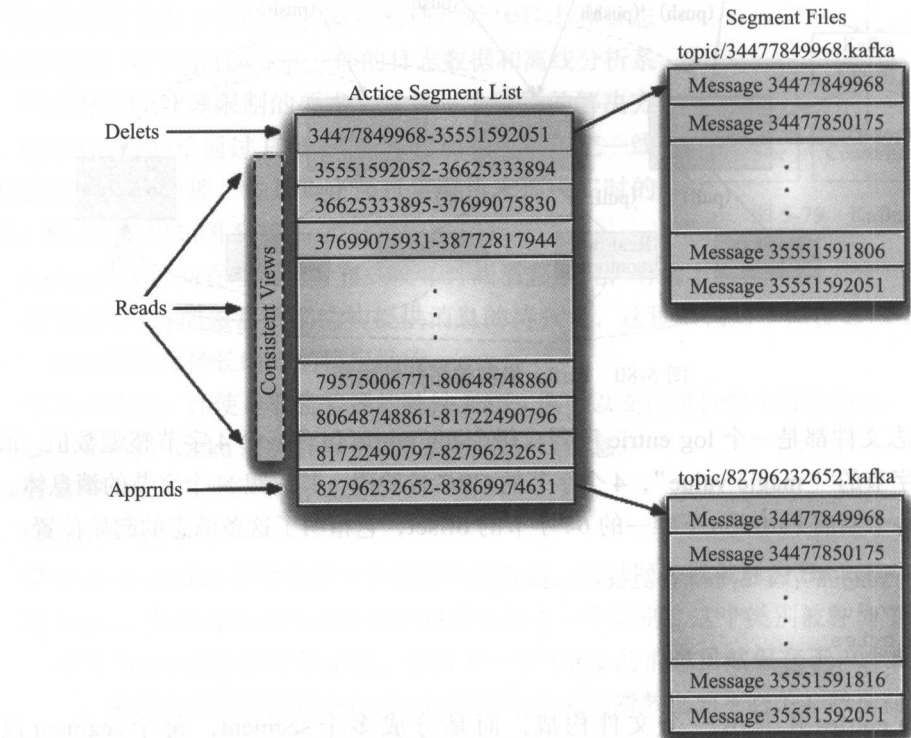


图 5-81 log entry 内容模式

这里要注意，因为 Kafka 读取特定消息的时间复杂度为 $O(1)$ ，即与文件大小无关，所以这里删除过期文件与提高 Kafka 性能无关。选择怎样的删除策略只与磁盘以及具体的需求有关。另外，Kafka 会为每一个 Consumer Group 保留一些 metadata 信息——当前消费的消息的 position，即 offset。这个 offset 由 Consumer 控制。正常情况下，Consumer 会在消费完一条消息后递增该 offset。当然，Consumer 也可将 offset 设成一个较小的值，重新消费一些消息。因为 offset 由 Consumer 控制，所以 Kafka Broker 是无状态的，它不需要标记哪些消息被消费过，也不需要通过 broker 来保证同一个 Consumer Group 只有一个 Consumer 能消费某一条消息，因此也就不需要锁机制，这也为 Kafka 的高吞吐率提供了有力保障。

Producer 消息路由

Producer 发送消息到 Broker 时，会根据 Partition 机制选择将其存储到哪一个 Partition。如果 Partition 机制设置合理，所有消息可以均匀分布到不同的 Partition 里，这样就实现了负

载均衡。如果一个 Topic 对应一个文件，那这个文件所在的机器 I/O 将会成为这个 Topic 的性能瓶颈，而有了 Partition 后，不同的消息可以并行写入不同 Broker 的不同 Partition 里，极大地提高了吞吐率。可以在 \$KAFKA_HOME/config/server.properties 中通过配置项 num.partitions 来指定新建 Topic 的默认 Partition 数量，也可在创建 Topic 时通过参数指定，同时也可以在该 Topic 创建之后通过 Kafka 提供的工具修改。

在发送一条消息时，可以指定这条消息的 key，Producer 根据这个 key 和 Partition 机制来判断应该将这条消息发送到哪个 Partition。Partition 机制可以通过指定 Producer 的 partition.class 这一参数来指定，该 class 必须实现 kafka.producer.Partitioner 接口。

Consumer Group

使用 Consumer high level API 时，同一 Topic 的一条消息只能被同一个 Consumer Group 内的一个 Consumer 消费，但多个 Consumer Group 可同时消费这一消息，如图 5-82 所示。

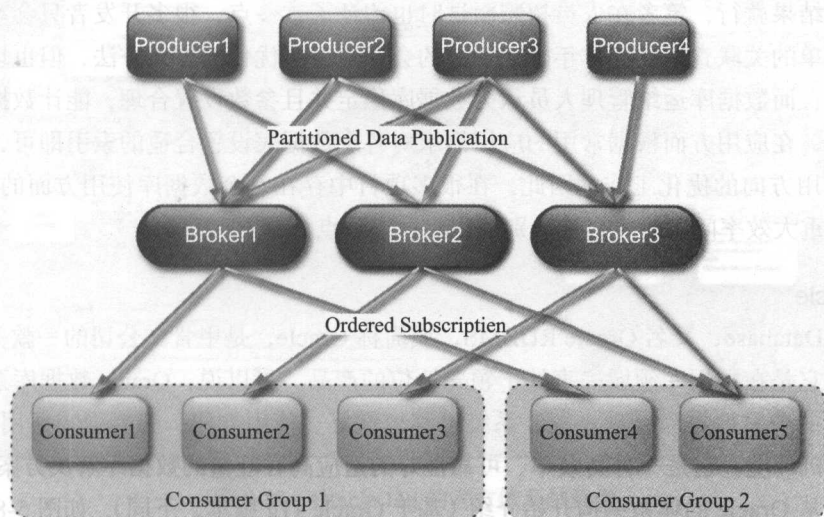


图 5-82 Consumer high level API 服务模式

这是 Kafka 用来实现一个 Topic 消息的广播（发给所有的 Consumer）和单播（发给某一个 Consumer）的手段。一个 Topic 可以对应多个 Consumer Group。如果需实现广播，只要每个 Consumer 有一个独立的 Group 即可。要实现单播只要所有的 Consumer 在同一个 Group 里。用 Consumer Group 还可以将 Consumer 进行自由的分组而不需要多次发送消息到不同的 Topic。

实际上，Kafka 的设计理念之一就是同时提供离线处理和实时处理。根据这一特性，可以使用 Storm 这种实时流处理系统对消息进行实时在线处理，同时使用 Hadoop 这种批处理系统进行离线处理，还可以同时将数据实时备份到另一个数据中心，只需要保证这三个操作所使用的 Consumer 属于不同的 Consumer Group 即可。

5.2.4 关系型数据库

关系型数据库应该算是当今世界上应用最为广泛的数据存储形式。关系型数据库的起源来自于 1970 年 IBM 的研究员 E.F.Codd 博士在刊物 *Communication of the ACM* 上发表了一篇名为 “*A Relational Model of Data for Large Shared Data Banks*” 的论文，提出了关系模型的概念，奠定了关系模型的理论基础。之后的 1979 年 Oracle（原名“关系式软件公司”（RSI））推出它的第一款商用的关系型数据库。关系型数据库真正发展是从 20 世纪 80 年代，关系型数据库的出现是数据库发展史上的一个巨大转折，在短短几十年间的发展，逐步成熟应对众多的数据应用需求，典型的两大分类为 OLTP（联机事务处理）和 OLAP（联机分析处理），关系型数据库依然成为众多系统不可或缺的组成部分。

但是，在很多项目的开发过程中存在一种较为普遍的现象，技术开发人员在开发过程中性能方面的考虑过多地放在代码层面，他们认为数据库就是一个黑盒子，只要能根据查询语句返回正确结果就行，笔者在人员招聘面试时也印证了这一点，很多开发者只会写一些单表操作以及简单的关联查询，工作年限比较久的会知道一些优化的方式方法，但也坦言说工作中很少用过；而数据库运维管理人员认为数据库稳定并且参数设置合理，能让数据库发挥它的最大性能，在应用方面根据常用的应用需求对各个数据表设置合适的索引即可，也很少涉及数据库使用方向的优化工作。因此，在很多项目中存在一个数据库使用方面的灰色地带，只是在出现重大效率问题时才会被个别关注。这里重点介绍 Oracle。

1. Oracle

Oracle Database，又名 Oracle RDBMS，或简称 Oracle，是甲骨文公司的一款关系数据库管理系统。它是在数据库领域一直处于领先地位的产品。可以说，Oracle 数据库系统是目前世界上流行的关系数据库管理系统，系统可移植性好、使用方便、功能强，适用于各类大、中、小、微机环境。它是一种高效率、可靠性好的适应高吞吐量的数据库解决方案。

我们先从 Oracle 的体系结构开始介绍（基于 Oracle 11g 版本，下同），如图 5-83 所示。

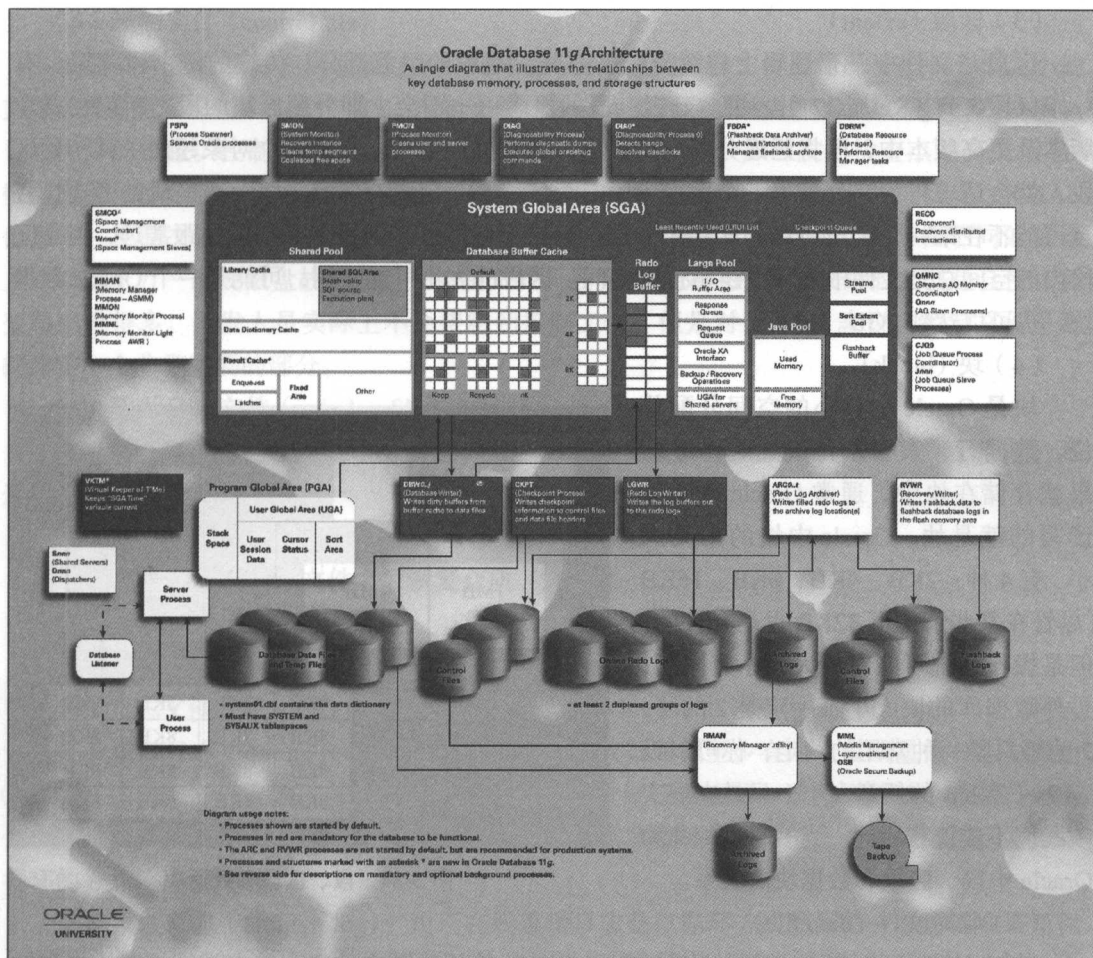
图 5-85 主要体现了 Oracle 体系结构的三大部分：

- 文件（桶状图标）。
- 内存结构（两大圆角矩形：SGA 和 PGA）。
- 物理进程或线程（尖角矩形图标）。

Oracle 文件

数据文件（data file）：这些文件是数据库的主要文件，其中包括数据表、索引和所有其他的段。

在介绍数据文件之前，我们需要先了解如何组织这些文件，以及在文件中如何组织数据。要了解这些内容，需要知道什么是表空间（tablespace）、什么是段（segment）、什么是区段（extent），以及什么是块（block）。这些都是 Oracle 在数据库中的存储对象所用的分配单位。



（3）区段（extent）

区段是文件中一个逻辑上连续分配的空间，是段的组成部分。传统的每个段都至少有一个区段，11g Release2 引入了“延迟”段的概念——不会立即分配区块的段，所以在这个版本及更高版本中，段将延迟分配初始区段，直到插入数据。如果一个对象超出了其初始区段，就会请求再为它分配另一个区段。第二个区段不一定就在磁盘上第一个区段的旁边，甚至可能不在第一个区段所在的文件中分配。第二个区段可能与第一个区段相距甚远，但是区段内的空间总是文件中的一个逻辑连续空间。区段的大小可能不同，可能是一个 Oracle 数据块，也可以达到 2GB。

（4）块（block）

块是 Oracle 中最小的空间分配单位。数据行、索引条目或临时排序结果就存储在块中。通常 Oracle 从磁盘读写的就是块。Oracle 中块的常见大小有 4 种：2KB、4KB、8KB、16KB（尽管在某些情况下 32KB 也是允许的，但是操作系统可能对最大大小有限制；另外数据库的块大小不一定是 2 的幂，2 的幂只是一种常用的约定，你可以设定 2 ~ 32KB 的任意大小，只是在现实中我们不建议利用这一点）。图 5-84 为 Oracle 中段、区段和数据块的关系。

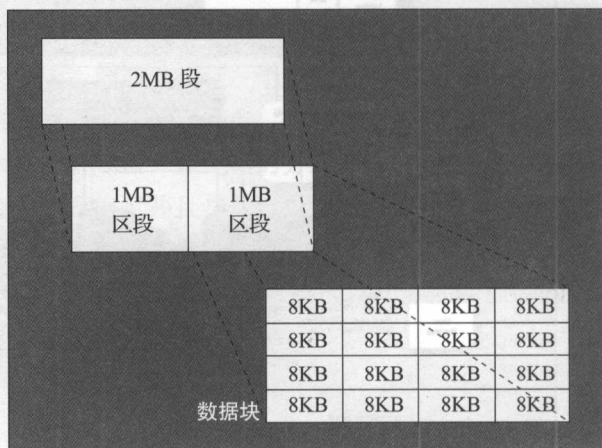


图 5-84 段、区段和数据块

（5）临时文件（data file）

临时文件是一种特殊类型的数据文件，Oracle 使用临时文件来存储大规模排序操作和队列操作的中间结果，如果 RAM 中没有足够的空间，还会用临时文件存储全局临时表数据或结果集数据。永久数据对象（例如表或者索引）不会存储在临时文件中，但是临时表及其索引的内容要存储在临时文件中。所以，不能在临时文件中创建表，但是使用临时表时完全可以在其中存储数据。关于真正的临时文件，有一个细节需要注意，如果操作系统允许创建临时文件，则会稀疏地创建，也就是说在需要之前它们不会真正占用磁盘存储空间。

那么问题来了，假如 DBA 通过命令创建了 2GB 的临时表空间，创建时底层操作系统有足够的空闲空间，但是随着数据库的数据膨胀导致操作系统剩余空间不足 2GB 之后，我们在使用临时文件存储临时数据时，就会得到“没有更多空间”的错误，从而存在安全隐患。解决该问题的方法为，在系统中创建一个非稀疏的文件，将原有的文件数据复制到该文件中，并使用 REUSE 选项利用该临时文件创建临时表空间，这样这个临时文件已经分配了所有的文件系统空间，而且数据库中确实有了 2GB 的临时空间可使用（这个问题的解决因操作系统而异）。

(6) 控制文件 (control file)

控制文件是 Oracle 在启动数据库实例时所需要的重要文件，一般情况下是一个相当小的文件（最多能增长到 64MB 左右）。控制文件在数据库实例启动时告知实例数据库和在线重做日志文件的位置。控制文件还会告知 Oracle 其他的一些事情，如已发生检查点的有关信息、数据库名（必须与 `db_name` 参数匹配）、创建数据库的时间戳、归档重做日志的历史、RMAN 信息等。控制文件应该通过硬件（RAID）多路保存，创建多个副本并且保存在不同的磁盘上，以防止万一出现磁盘故障而丢失控制文件。丢失控制文件并不是致命的，但是会使恢复变得困难得多。开发人员实际上不会接触到控制文件，但是对于 DBA 来说，控制文件是数据库中一个非常重要的部分。

(7) 重做日志文件 (redo log file)

重做日志文件是数据库的事务日志，对于 Oracle 数据库至关重要，通常只用于恢复，不过也可以用于以下工作：

- ❑ 系统崩溃后的实例恢复。
- ❑ 通过备份恢复数据文件之后恢复介质。
- ❑ 备用 (standby) 数据库处理。
- ❑ 输入到流中，用于实现信息共享。

重做日志文件只要分为在线重做日志 (online redo log) 和归档重做日志 (archive log) 两类。

在线重做日志 (online redo log)：每个 Oracle 数据库都至少有两个在线重做日志文件组（通常为三个）。每个重做日志组都包括一个或多个重做日志成员。这些组的单个重做日志文件成员之间实际上是形成彼此真正的镜像。这些在线重做日志文件的大小是固定的，并且以循环的方式使用。

归档重做日志 (archive log)：由于在线重做日志是以循环方式使用的，在所有在线重做日志文件组都已经写满之后，会覆盖并重写最先写满的日志文件，从而丢失日志文件的数据，在这种情况下除非你保留了这个文件，否则数据库将无法从备份将数据恢复到当前的时间点。归档重做日志是用于保留所有的数据库重做日志数据，该文件只能在 Oracle 数据库采用 ARCHIVELOG 模式下产生。

还有一类日志文件为闪回日志文件 (flashback log)，是 Oracle 10g 的新特性。闪回日志包括已修改数据库块的“前映像”，可用于将数据库返回（恢复）到该时间点之前的状态。要使用这个特性，数据库必须采用 ARCHIVELOG 模式，并且必须配置为支持 FLASHBACK DATABASE 命令。我的意思是，在你使用这个功能之前，必须先行配置，等到真正发生了破坏，再想启动这个功能就为时过晚了，使用时必须早做打算。

Oracle 除了上面介绍的三大类重要的文件以外还有一些其他相关文件：

(1) 参数文件 (parameter files)

参数文件定义了实例的特性，Oracle 实例在启动到 nomount 状态时需要加载参数文件，读取参数文件中相关的初始化配置信息，例如系统全局区中的内存结构大小、DBWR 的个

数等，最主要的是指定了控制文件所在路径，Oracle 实例在从 nomount 状态启动到 mount 状态时需要加载控制文件。参数文件在 Oracle 系统中分为两种，一种是文本类型的参数文件 pfile；另一种是二进制的参数文件 spfile。

(2) 口令文件 (password files)

口令文件主要用于通过网络完成管理活动的用户进行认证。即使数据库实例在未启动的情况下，上面的这些用户也可以通过密码验证进行数据库的连接。

Oracle 内存结构

Oracle 的内存结构主要分两类：

❑ 系统全局区 (System Global Area, SGA)：这是一个很大的共享内存段，几乎所有的 Oracle 进程都要访问这个区中的某一点。

❑ 进程全局区 (Process Global Area, PGA)：这是一个进程或线程专用的内存，其他进程 / 线程不能访问。

(1) 系统全局区 (SGA)

SGA 主要包括共享池 (Shared Pool)、数据库高速缓存 (Database Buffer Cache)，系统重做日志缓存 (Redo Log Buffer) 三部分，以及其他方面的信息。SGA 结构如图 5-85 所示。

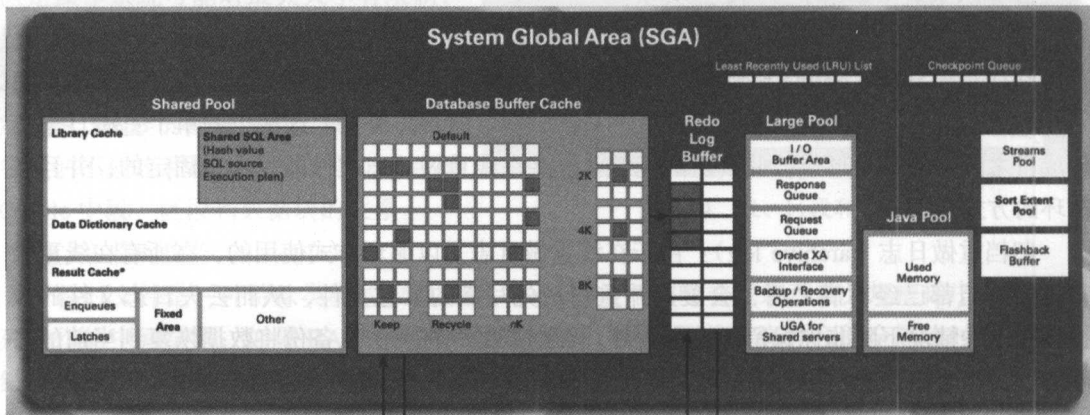


图 5-85 SGA 结构图

1) Shared Pool：缓存了各用户间可共享的各种结构。Oracle 通过 Shared Pool 来实现 SQL 共享、减少代码硬解析等从而提高数据库性能。

其中主要由两部分组成，一部分是库缓存 (Library Cache)，另一部分是数据字典缓存 (Data Dictionary Cache)。Library Cache 主要用于存储 SQL 语句、SQL 语句相关解析树、执行计划、PL/SQL 程序块 (包括匿名程序块、存储过程、包、函数等) 以及它们转换后能够被 Oracle 执行的代码等，这部分信息可以通过 v\$librarycache 视图来查询；Dictionary Cache 主要用于存放数据字典信息，包括表、视图等对象的结构信息，用户以及对象权限信息，这部分信息相对稳定，可以通过 v\$rowcache 进行查看。

2) Database Buffer Cache: 是临时存放数据的共享内存, 缓存了从磁盘上检索的数据块。

举例说明: 用户提交了查询语句 `select * from emp` 的请求, Oracle 会首先查看 Database Buffer Cache 中有没有 `select * from emp` 查询过的数据, 如果有则直接从 Database Buffer Cache 中读取返回给客户; 如果没有则直接操作硬盘读写, 把数据读写到 Database Buffer Cache 中, 再返回给客户。

Database Buffer Cache 的内容主要分成三种类型: Default、Keep、Recycle, 在 8i 之前这三种类型是独立指定的, 互不制约; 在 8i 之后三者的大小相加就是整体的 `db_cache_size`。比如, 指定了 Keep 和 Recycle 的 Buffer Cache, 则 Default 类型的 Buffer Cache 的大小就是 `db_cache_size - buffer_pool_keep - buffer_pool_recycle`。

这里要说明的是, 从名字上看, 很容易让人误以为这三种 Buffer Cache 是三种不同的治理内存数据库的机制, 但实际上, 它们之前的治理和内部机制上没有任何的区别。仅仅是为 DBA 提供一个选择, 能够将数据库对象分成“非常热的”“比较热的”和“不热的”这三种类型。频繁被访问的数据, 为防止这些对象被清除出内存, 将这些对象移入 Keep Buffer Cache, 偶尔被访问或者说不频繁被访问的数据, 将这些对象移入 Recycle Buffer Cache, 这样以保证更多的内存可提供给 Oracle 进行分配, 即使偶然被查询只需从磁盘中再读取一次。

9i 之后 Oracle 还提供了可以设置多种数据块尺寸 (2KB、4KB、8KB、16KB 或者 32KB) 的 Buffer Cache, 以便存放不同数据块尺寸的表空间对象, 称之为 `nk buffer cache`, 初始化参数为 `db_nk_cache_size` 用于指定不同数据块尺寸, 这里的 `n` 就是 2、4、8、16 或者 32。

3) Redo Log Buffer: 数据在写到在线重做日志之前, 会在重做日志缓冲区中临时存储这些数据。重做日志缓冲区主要是为了加快数据的操作。但是数据在重做日志缓冲区中不会停留太久, 会在以下几种情况发生时对这个区域进行刷新写入到在线重做日志:

- ☐ 每 3 秒一次。
- ☐ 执行提交操作。
- ☐ 切换在线重做日志。
- ☐ 重做日志缓冲区满 1/3, 或者缓冲数据超过 1MB。

4) 其他:

Large Pool: 一个可选的区域, 用来缓存大的 I/O 请求, 以支持并行查询、共享服务器模式以及某些备份操作。

☐ **Java Pool:** 保存 Java 虚拟机中特定会话的数据与 Java 代码。在 Oracle 10g 中, Java 池可以在数据库启动并运行在线调整大小。

☐ **Streams Pool:** 由 Oracle streams 使用。

☐ **Sort Extent Pool:** 用于所有活动排序段和回滚段中的扩展。

☐ **Flashback Buffer:** 当 Oracle 启动 flashback database 的情况下, SGA 中开辟一块新区域, 用于缓存闪回日志。

(2) 进程全局区 (PGA)

PGA 与 SGA 类似，都是 Oracle 系统为会话在服务器内存中分配的区域，只不过两者的作用不同，共享程度也不同。PGA 主要是为了某个用户进程所服务的，这个内存区不是共享的，只有这个用户的服务进程本身才能够访问它自己的 PGA 区。

PGA 主要包括以下几个部分：

❑ Stack Space：包括其他的会员变量。

❑ User Session Data：包括会话的用户权限和优化统计信息。

❑ Cursor Status：用于指示会话当前所使用的 SQL 语言的处理状态。

❑ Sort Area：用于处理 SQL 语句（例如排序查询、表的有限制关联、创建索引等）的排序操作提供所需内存空间。如果排序所需的内存空间大于 Sort Area 的设置，则排序操作也会使用临时表空间的磁盘来进行。

Oracle 物理进程与线程

Oracle 的物理进程与线程主要分为服务器进程（Server Process）、后台进程（Background Process）以及一些从属进程（Slave Process）。

这里我们只介绍一些核心的后台进程，这些进程会随着数据库的启动而启动，用于完成各种数据库的维护工作，例如服务监控、重做日志的维护、数据保存到数据文件等。

PMON：进程监视器，主要负责三件事。一是清理异常中止的连接；二是监控其他后台进程，并在有必要的情况下重启这些后台进程；三是向 Oracle TNS 监听器注册当前数据库实例。

SMON：系统监视器，与进程监视器作用类似，但 SMON 主要是完成系统级别的维护工作，例如清理临时空间、RAC 中失败节点的实例恢复、收缩回滚段等。

CKPT：检查点进程，这个检查点进程的作用不是在于创建检查点，而是通过更新数据文件的文件头信息，来辅助真正创建检查点的进程（DBWn）。

DBWn：数据库块写入器，总体来说这个进程就是负责将缓冲区中的数据脏块写入磁盘的后台进程。可以配置多个 DBWn，主要是为了保证足够的性能，能够短时间内将脏块写入磁盘，并释放缓冲区，而利用多个 DBWn 来分布工作负载。

LGWR：日志写入器，负责将 SGA 中重做日志缓冲区的内容刷新输出到磁盘。

ARCn：归档进程，负责在 LGWR 将在线重做日志写满时，将其复制到另一个位置，作为归档日志存储。

FBDA：闪回数据归档进程，这是 Oracle Database 11g 版本新增的后台进程，是新增的闪回数据归档功能的重要组成部分。

DBRM：数据库资源管理进程，会为一个数据库实例设置指定的资源计划，完成与实施 / 实现这些资源计划相关的各种操作。

GEN0：通用任务执行进程，为数据库提供一个通用的任务执行线程，主要目标是分担另外某个进程的阻塞处理，并在后台完成所阻塞的操作。

2. MySQL

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，2008 年 1 月 MySQL 被 Sun 公司收购，2009 年 4 月 Oracle 收购了 Sun 公司，MySQL 转入 Oracle 门下。目前 MySQL 被广泛地应用在 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站的数据库。

MySQL 主要是由连接池组件、管理服务和工具组件、SQL 接口、解析器、优化器、缓存、存储引擎、物理文件几个部分组成。图 5-86 为 MySQL 的体系结构图。

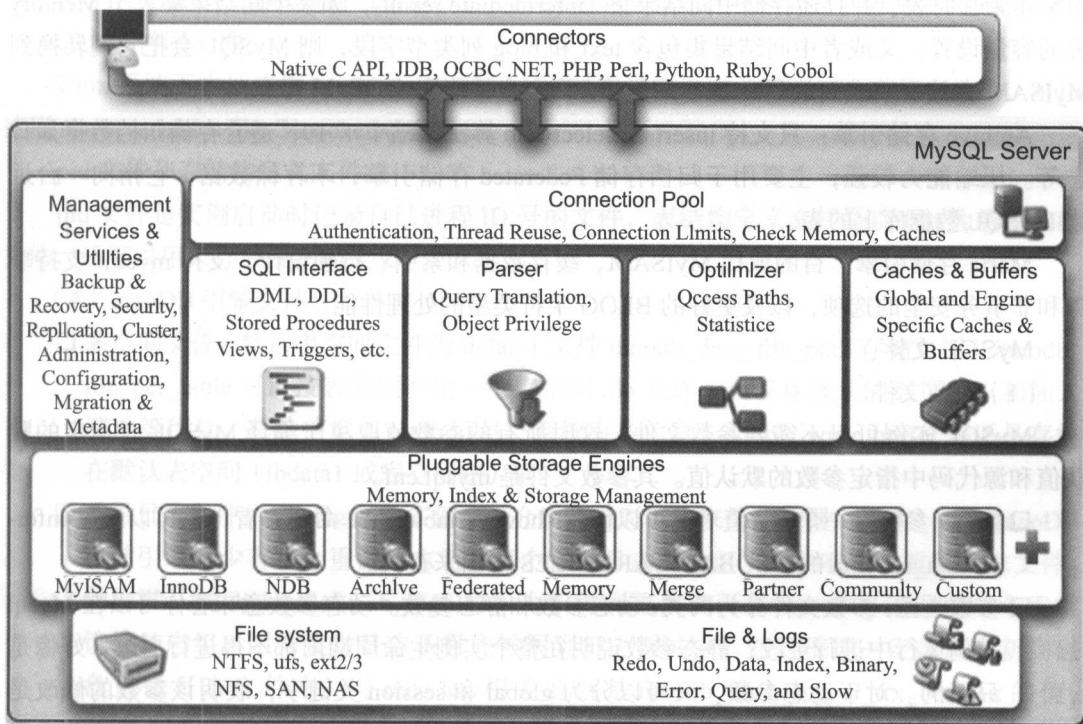


图 5-86 MySQL 的体系结构图

MySQL 各个存储引擎概述

InnoDB 存储引擎: 面向 OLTP (Online Transaction Processing)、行锁、支持外键、非锁定读，MySQL-4.1 开始支持每个 InnoDB 引擎的表单独放到一个表空间里。InnoDB 通过使用 MVCC 来获取高并发性，并且实现 SQL 标准的 4 种隔离级别，同时使用一种被称成 next-key locking 的策略来避免幻读 (phantom) 现象。除此之外，InnoDB 引擎还提供了插入缓存 (insert buffer)、二次写 (double write)、自适应哈希索引 (adaptive hash index)、预读 (read ahead) 等高性能技术。

MyISAM 存储引擎: 不支持事务、表锁、全文索引、适合 olap (在线分析处理) 应用，其中 myd: 放数据文件，myi: 放索引文件。MySQL-5.0 版本之前，MyISAM 默认支持的表大小为

4G，从 MySQL-5.0 以后，MyISAM 默认支持 256T 的表数据。MyISAM 只缓存索引数据。

NDB 存储引擎：集群存储引擎，share nothing，特点是数据放在内存中，可提高可用性。MySQL-5.1 版本开始可以将非索引数据放到磁盘上。NDB 之前的缺陷是 join 查询是 MySQL 数据库层完成的，而不是存储引擎完成的，复杂的 join 查询需要巨大的网络开销，速度很慢。当前 MySQL cluster 7.2 版本中已经解决此问题，join 查询效率提高了 70 倍。

Memory 存储引擎：数据存放在内存中，表锁，并发性能差，默认使用哈希索引，不支持 text 和 blob 类型，varchar 是按照 char 的方式来存储的。MySQL 数据库使用 Memory 存储引擎作为临时表，并且还存储中间结果集（intermediate result），如果中间结果集大于 Memory 表的容量设置，又或者中间结果集包含 text 和 blob 列类型字段，则 MySQL 会把它们转换到 MyISAM 存储引擎表而放到磁盘上，会对查询产生性能影响。

Archive 存储引擎：只支持 insert 和 select 且 zlib 算法压缩 1 : 10，适合存储归档数据如日志等。压缩能力较强，主要用于归档存储 Federated 存储引擎，不存储数据，它指向一台远程 MySQL 数据库上的表。

Maria 存储引擎：目的取代 MyISAM、缓存数据和索引、行锁设计、支持 mvcc，支持事务和非事务安全的选项，以及更好的 BLOG 字符类型的处理性能。

MySQL 文件

（1）参数文件

MySQL 实例可以不需要参数文件，这时所有的参数值取决于编译 MySQL 时指定的默认值和源代码中指定参数的默认值。其参数文件是 mysql.cnf。

❑ **参数：**参数是一个键 / 值对。可以使用 show variables like 命令查看，也可以通过 information_schema 的 GLOBAL_VARIABLES 视图来查找。

❑ **参数类型：**参数文件分为两类：动态参数和静态参数。动态参数意味着你可以在 Mysql 实例运行中进行更改；静态参数说明在整个实例生命周期内都不得进行更改，好像是只读的。对于动态参数，又可以分为 global 和 session 关键字，表明该参数的修改是基于当前会话还是整个实例的生命周期。有些动态参数只能在会话中进行修改，例如 autocommit；有些参数修改完后，在整个实例生命周期中都会生效，例如 binlog_cache_size；而有些参数既可以在会话又可以在整个实例的生命周期内生效，例如 read_buffer_size。

（2）日志文件

❑ **错误日志：**错误日志对 MySQL 的启动、运行、关闭过程进行了记录。出现 MySQL 不能正常启动时，第一个必须查找的文件应该就是错误日志文件。使用 show variables like 'log_error' 来定位文件。

❑ **慢查询日志：**慢查询能为 SQL 语句的优化带来很好的帮助。设定一个阈值，将运行时间超过该值的所有 SQL 语句都记录到慢查询日志文件中。用参数 long_query_time 来设置。另一个和慢查询日志有关的参数是 log_queries_not_using_indexes，若运行的

SQL 语句没有使用索引, 则这条 SQL 语句会被记录下来。

- 查询日志: 查询日志记录了所有对 MySQL 请求的信息, 不论这些请求是否得到正确的执行。默认文件名为: 主机名 .log。
- 二进制日志: 二进制记录了对数据库执行更改的所有操作, 但是不包括 SELECT 和 SHOW 操作, 还包括了执行时间和更改操作时间。可用来恢复某些数据, 同时也可以用来复制同步远程数据库。将 binlog_format 设置成 row, 可以支持事务隔离级别为 READ COMMITTED, 以获得更好的并发性。在使用 MIXED 格式下, MySQL 采用 STATEMENT 格式进行二进制日志文件的记录, 但是有一些情况下会使用 ROW 格式。

(3) 套件字文件

Unix 系统下本地连接 MySQL 可以采用 UNIX 套接字方法, 需要一个套接字文件, 可以使用 show variables like '%socket%' 查询。

pid 文件和表结构定义文件

pid 文件是实例启动时记录自己进程 ID 号的文件, 表结构定义文件是以 frm 为后缀名的文件, 还可以用来存放视图的定义。

(4) InnoDB 引擎文件

- 表空间文件: 默认表空间文件为 ibdata1 文件 innodb_data_file_path 存储数据, innodb_file_per_table 可以按表分别产生一个表空间 .db 文件, 但仅存该表的数据索引和插入缓冲等信息, 其他信息如 undo 信息、系统事务信息、double write buffer 等还是存放在默认表空间 (ibdata1 或表空间组) 里。
- 重做日志文件: redo log 是在实例或者介质失败时, 用来保证数据完整性。每个 InnoDB 存储引擎至少有一个重做日志组, 每个重做日志文件组下至少有 2 个重做日志文件, 例如默认的 ib_logfile0、ib_logfile1。为了得到更高的可靠性, 你可以设置多个重做镜像日志组。因为重做日志条目先被写到日志缓冲中, 然后根据一定条件刷新到磁盘重做日志文件中。与 redo log 相关的就是 innodb_flush_log_at_trx_commit 的值, 对 InnoDB 的性能影响很大。它有 0、1、2 三个值, 0 代表提交事务时, 并不同步写 redo log, 而是等 master thread 每秒一次写入; 1 代表 commit 时就将 redo log 缓存写入磁盘; 2 代表 commit 时将 redo log 缓存异步地写入磁盘。

5.2.5 数据 ETL

数据 ETL, 主要是表述数据处理流程的三个操作 (ETL, Extract-Transform-Load)。

- 抽取 (Extract): 一般抽取过程需要支持连接到不同的数据源获取数据的功能, 这部分操作看似简单, 实际上它是 ETL 解决方案成功实施的一个主要障碍。
- 转换 (Transform): 任何对数据的处理过程都算是转换, 包括数据移动、规则校验、数据内容或格式的修改、数据计算等。
- 加载 (Load): 把数据加载的目标系统。

1. Kettle

Kettle 介绍

Kettle 是一款纯 Java 编写的开源数据 ETL 工具，可支持在多种操作系统上运行，数据抽取高效稳定。在 2006 年 Kettle 加入了开源组织 Pentaho 正式命名为 PDI，英文全称为 Pentaho Data Integration。这里需要注意的是，Kettle 在 4.3 之前的版本 License 为 LGPL 的，也就是说基于这些版本进行二次开发出的产品也必须进行开源，在 2012 年 Kettle 将 License 改为 Apache2。

官方文档：<https://help.pentaho.com/Documentation/6.1>。

官方论坛：<http://forums.pentaho.org/forumdisplay.php?f=135>。

中文论坛：<http://www.pentahochina.com>。

Bug 报告地址：<http://jira.pentaho.com/browse/PDI>。

当前版本：Version 6.1（2016 年）。

Kettle 的 3 个子程序：Spoon、Pan、Kitchen。

□ Spoon 允许你通过图形界面来设计 ETL 转换过程（Transformation）。

□ Pan 允许你批量运行由 Spoon 设计的 ETL 转换（例如使用一个时间调度器）。Pan 是一个后台执行的程序，没有图形界面。

□ Kitchen 允许你批量使用由 Chef 设计的任务（例如使用一个时间调度器）。Kitchen 也是一个后台运行的程序。

Kettle 的术语

Transformation：转换，可以理解为将一个或者多个不同的数据源组装成一条流水线。然后最终输出到某一个地方、文件或者数据库等。如图 5-87 所示为 Kettle 流水线工作模式。

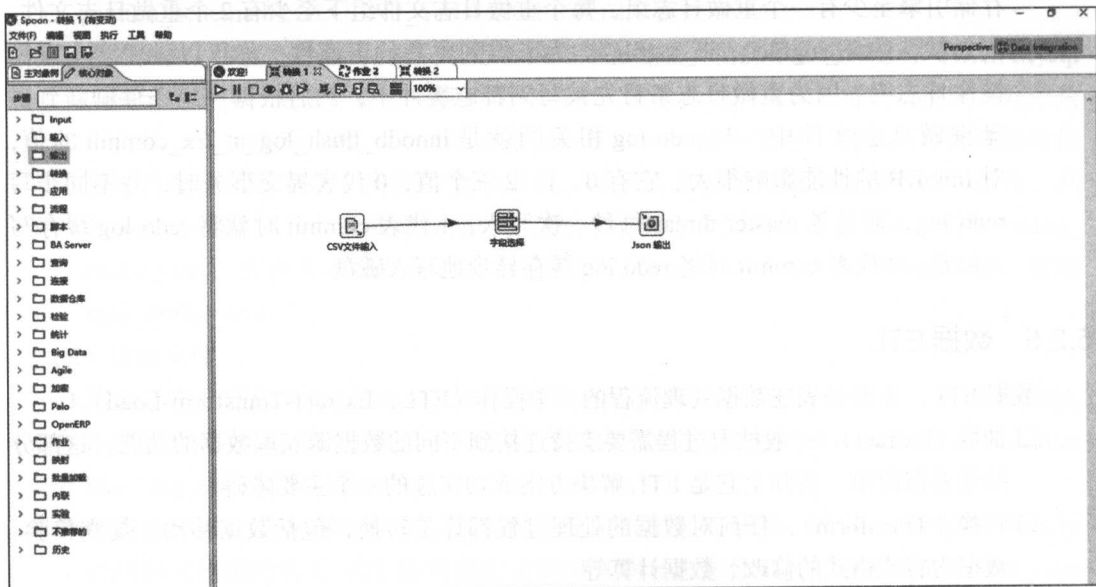


图 5-87 Kettle 流水线工作模式

Job：作业，可以调度设计好的转换，也可以执行一些文件处理（比较、删除），还可以上传下载文件、发送邮件、执行 Shell 命令等。

Job 与 Transformation 的差别是：Transformation 专注于数据的 ETL，而 Job 的范围比较广，可以是 Transformation，也可以是 Mail、SQL、Shell、FTP 等，甚至可以是另外一个 Job。如图 5-88 所示为一个 Job。

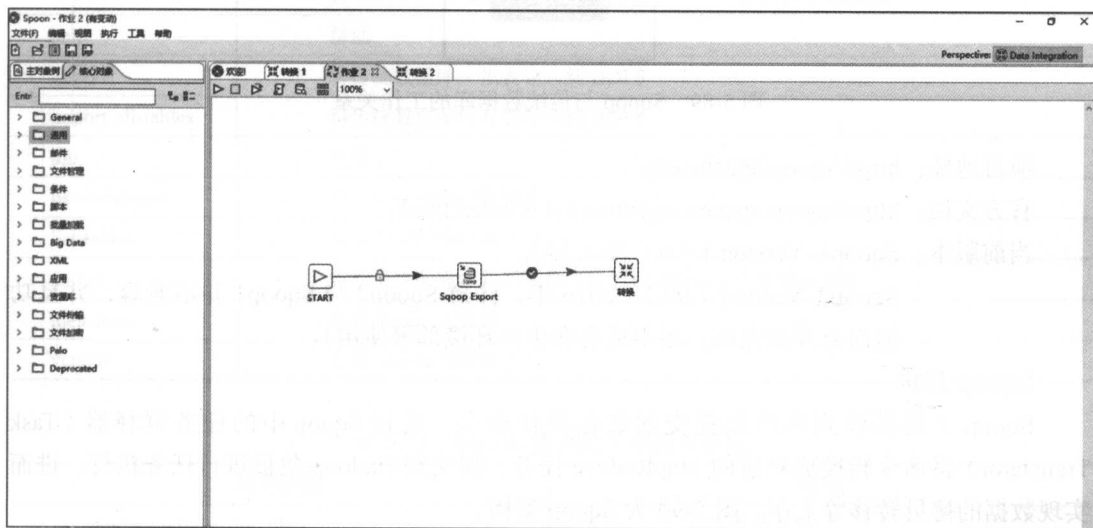


图 5-88 Job 工作

Hop：连接 Transformation 步骤或者连接 Job（实际上就是执行顺序）的连线。

Transformation hop：主要表示数据的流向，从输入、过滤等转换操作到输出。

Job hop：可设置执行条件：无条件执行、当上一个 Job 执行结果为 true 时执行、当上一个 Job 执行结果为 false 时执行。

Kettle 日志级别说明

- ☐ Nothing：不记录任何信息，执行效率最高。
- ☐ Error：只记录错误信息。
- ☐ Minimal：记录最少的信息。
- ☐ Basic：记录基本信息。
- ☐ Detailed：记录详细信息。
- ☐ Debug：记录调试信息。
- ☐ Rowlevel：转换过程中的每一行都记录下来，日志最详细，执行效率最低。

2. Sqoop

Sqoop 介绍

Sqoop 是一款开源的数据同步工具，主要用于 Hadoop（Hive、HBase）与传统数据库之

间进行相互的数据同步工作。图 5-89 显示了 Sqoop 与传统数据库的工作关系。

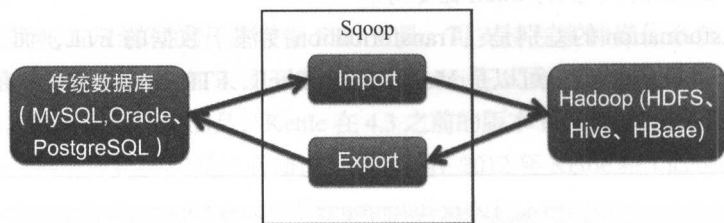


图 5-89 Sqoop 与传统数据库的工作关系

项目地址：<http://sqoop.apache.org/>。

官方文档：<http://sqoop.apache.org/docs/1.4.6/index.html>。

当前版本：Sqoop1-Version 1.4.6（2016 年）；

Sqoop2-Version 1.99.7（2016 年，注意 Sqoop2 与 Sqoop1 并不兼容，并且功能尚未开发完成，还不适合在生产环境部署使用）。

Sqoop 架构

Sqoop 工具接收到客户端提交的数据操作命令，通过 Sqoop 中的任务解释器（Task Translator）将命令转换成对应的 MapReduce 任务，提交到 Hadoop 集群进行任务执行，进而实现数据的拷贝转移等工作。图 5-90 为 Sqoop 架构。

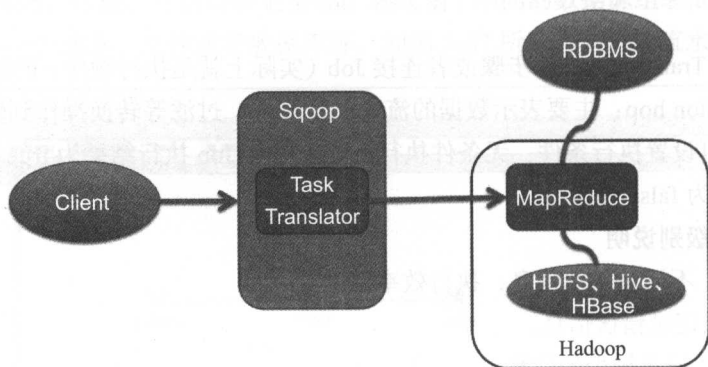


图 5-90 Sqoop 架构



由于 Sqoop 最终是将任务转换成 MapReduce 任务提交到 Hadoop 集群执行，所以 Hadoop 集群的相关节点所在服务器需要跟源端 / 目标端之间的网络是可通的。

Sqoop 命令

Sqoop 现阶段提供 13 种操作命令和 7 个通用的参数，如表 5-14 和表 5-15 所示。

表 5-14 Sqoop 提供的 13 种操作命令

操作命令	说明
codegen	将关系数据库表映射为一个 Java 文件、Java Class 类，以及相关的 jar 包
create-hive-table	导入表定义到 Hive
eval	执行 SQL 查询语句，并将结果显示
export	将 HDFS 路径导出到传统数据库表
help	帮助
import	将传统数据库一张表导入到 HDFS
import-all-tables	将传统数据库所有表导入到 HDFS
job	任务
list-databases	列出所有可用的数据库
list-tables	列出所有可用的表
merge	将两个数据集合并的工具，对于相同的 key 会覆盖老的值
metastore	查看 Sqoop 元数据
version	查看 Sqoop 当前版本号

表 5-15 Sqoop 提供的 7 个通用参数

通用参数	说明
-conf <configuration file>	指定参数文件
-D <property=value>	指定特定参数
-fs <local namenode:port>	指定 namenode 地址
-jt <local resourcemanager:port>	指定 ResourceManager 地址
-files <comma separated list of files>	指定要复制的文件
-libjars <comma separated list of jars>	指定需加载的 jar 包文件
-archives <comma separated list of archives>	指定归档

Sqoop 常用命令及常用参数介绍如表 5-16 所示。

表 5-16 Sqoop 常用命令和参数

命令	解释	常用的参数列表
list-databases	列出所有数据库	--connect <jdbc-uri> ##jdbc 连接字符串 --driver <class-name> ## JDBC 驱动类 --password <password> ## 密码 --username <username> ## 用户名
list-tables	列出所有表	--connect <jdbc-uri> ##jdbc 连接字符串 --driver <class-name> ## JDBC 驱动类 --password <password> ## 密码 --username <username> ## 用户名

(续)

命令	解释	常用的参数列表
import	将数据从关系数据库导入文件到Hive表中	<pre>--connect <jdbc-uri> ##jdbc 连接字符串 --driver <class-name> ## JDBC 驱动类 --password <password> ## 密码 --username <username> ## 用户名 --table ##hive 表名 --columns <col, col, col...> ## 导入指定列 --where <where clause> ## 数据导入条件 --hive-import ## 导入 hive --hive-table ## 导入 hive 后的表名 --append ## 使用追加模式 --delete-target-dir ## 导入前删除目标目录 --m ##map 任务数 --fields-terminated-by ## 列分隔符 --as-textfile ## 作为文本格式导入 --as-sequencefile ## 作为序列化文件导入 -z, --compress ## 指定导入压缩 --hive-delims-replacement <arg> ## 删除数据中包含的指定行列分隔符 --hive-drop-import-delims ## 删除数据中包含的所有行列分隔符 --null-non-string <null-str> ## 指定非字符类型 null 的表达式 --null-string <null-str> ## 指定字符类型 null 的表达式</pre>
export	将 Hive 中的表数据导入到 MySQL 数据库表中	<pre>--connect <jdbc-uri> ##jdbc 连接字符串 --driver <class-name> ## JDBC 驱动类 --password <password> ## 密码 --username <username> ## 用户名 --table ##hive 表名 --columns <col, col, col...> ## 导入指定列 --export-dir ## 导出 hive 路径 --m ##map 任务数 --input-fields-terminated-by ## 列分隔符 --input-lines-terminated-by ## 行分隔符 --mysql-delimiters ## 使用 mysql 默认分隔符 --null-non-string <null-str> ## 指定非字符类型 null 的表达式 --null-string <null-str> ## 指定字符类型 null 的表达式</pre>
help	查看 Sqoop 命令帮助	<pre>COMMAND ## 查看 sqoop 命令相关帮助</pre>

5.3 大数据算法库

随着大数据的日益发展，机器学习也成为目前大数据分析领域的一个热点内容。其实，在平时的学习和生活中也经常会用各种各样的机器学习算法，实际上基于 Python、Java 等的很多机器学习算法基本都被前人实现过很多次了。这些算法在网上可以找到很多，然而往往存在很多“脏”或者“乱”的开源代码。

在这样的背景下，我们总结了以下几个成熟稳定的开源算法库，这些开源库大多与垃圾邮件过滤、人脸识别、推荐引擎相关。它们大多数基于现今最流行的语言以及平台，推广以及扩展了机器学习领域的很多重要算法。从中，不但可以找到 LDA 等主题模型，也可以找到 HMM 等隐马尔可夫模型。这些模型都是应用领域的热点，也是研究者最需要的。

1. RHadoop

RHadoop 是 Revolution Analytics 下的一个开源库，它可以将统计语言 R 与 Hadoop 结合起来，与 Rhipe 类似，也可以在 MapReduce 模式下执行 R 函数。目前，该项目包括以下几个 R packages：plymr 包可以在 Hadoop 中对大数据集进行一些常用的数据整理操作。rmr 包提供了一些让 R 和 Hadoop 联合作业的函数。rdfs 包提供了一些函数来连接 R 和分布式文件系统 (HDFS)。rhbase 包中的函数则能连接 R 和 HBase。

其中，Hadoop 主要用来存储海量数据，R 语言完成 MapReduce 算法，用来替代 Java 的 MapReduce 实现。有了 RHadoop 可以让广大的 R 语言爱好者，有更强大的工具处理大数据。1GB、10GB、100GB、TB、PB 由于大数据所带来的单机性能问题，可能会一去不复返了。对于单独的 R 语言爱好者、Java 爱好者，或者 Hadoop 爱好者来说，同时具备三种语言知识并不容易。

RHadoop 的安装 (rhdfs 与 rmr2)

所谓的 RHadoop 安装并不是一个另外的单独的 software，而是 R package：rhdfs、rmr2、rhbase。从名字不难推断每个包的用途，rhdfs 是对应 HDFS 的 R interface，rmr2 是对应 MapReduce，rhbase 是对应 HBase。下载网址为 <https://github.com/RevolutionAnalytics/RHadoop/wiki/Downloads>，另外还需要从地址 (<https://cran.r-project.org/src/base/R-3/>) 下载 R 语言的 tar 包。需要注意的是，在安装 RHadoop 之前先安装并启动 Hadoop 环境，包括 hdfs、hbase 和 thrift 等，同时也建议安装 C++ 或 Python 库。

在 centos6.5 上安装 R，然后安装相关依赖包：

```
#yum install gcc-gfortran
#yum install gcc gcc-c++
#yum install readline-devel
#yum install libXt-devel
```

```
# tar xvf R-3.2.3.tar.gz
# cd R-3.2.3
# ./configure
# make
# make install
```

确认 Java 环境变量：RHadoop 依赖于 rJava 包，安装 rJava 前确认已经配置了 Java 环境变量，然后利用 R 对 jvm 建立连接。

```
[root@dataserver R-3.2.3]# cat /etc/profile 结尾添加
```

```
#####
```

```

export JAVA_HOME=/usr/java/jdk1.7.0_79
export JRE_HOME=/usr/java/jdk1.7.0_79/jre
export PATH=/bin: /usr/local/sbin: /usr/local/bin: /sbin: /bin: /usr/sbin: /usr/bin: /
root/bin
export CLASSPATH=.: /lib/dt.jar: /lib/tool.jar
export HADOOP_CMD=/usr/bin/hadoop
export HADOOP_STREAMING=/usr/hdp/current/hadoop-mapreduce-client/
hadoop-streaming.jar
export HADOOP_HOME=/usr/hdp/current/hadoop-client
export JAVA_HOME JRE_HOME PATH CLASSPATH
#####
[root@dataserver R-3.2.3]# R CMD javareconf

```

安装相关的依赖包，确保 RHadoop 软件包能正常使用。

```

[root@dataserver R-3.2.3]# R
> install.packages("rJava")
> install.packages("reshape2")
> install.packages("Rcpp")
> install.packages("iterators")
> install.packages("itertools")
> install.packages("digest")
> install.packages("RJSONIO")
> install.packages("functional")
> install.packages("bitops")
> install.packages("caTools")
> quit()

```

或者

```

install.packages(c("rJava", "Rcpp", "RJSONIO", "bitops", "digest", "functional",
"stringr", "plyr", "reshape2", "caTools"))

```

安装 RHadoop 软件包

```

[root@dataserver R-3.2.3]# export HADOOP_CMD=/usr/bin/hadoop
[root@dataserver R-3.2.3]# export HADOOP_STREAMING=/usr/hdp/current/
hadoop-mapreduce-client/hadoop-streaming.jar
[root@dataserver R-3.0.2]# R CMD INSTALL rhdfs_1.0.8.tar.gz
[root@dataserver R-3.0.2]# R CMD INSTALL rmr2_3.3.1.tar.gz
[root@dataserver R-3.0.2]# R CMD INSTALL rhbase_1.2.1.tar.gz

```

在以上过程中可能会产生一些问题，例如进入 R 之后用 library 载入 rhdfs 或者 rmr2 时发现系统提示不存在该库，这里遇到的问题有几点解决方法：

- ❑ R 语言目录并没有设置成当前的 /home/hadoop/software/R，而是最初系统默认的 /usr/local/R，导致 rhdfs、rJava、rmr2 以及那些依赖项都装到系统默认文件路径里了，所以需要做的第一件事是通过 cp 命令把默认路径中的 library 文件夹拷到目标路径 library 中。
- ❑ 如果当前用户没有 root 权限，进入 R 中有些操作是无法成功的，所以最好还是在 /etc/sudoer 中给自己的用户设上全部权限。

使用 RHadoop 软件包

```
[root@dataserver R-3.2.3]# R
> library(rhdfs)
> hdfs.init()
> hdfs.ls("/")

[root@dataserver R-3.2.3]# export HADOOP_HOME=/usr/hdp/current/
Hadoop-client
> library(rmr2)
```

MapReduce 的 R 语言程序

```
> small.ints = to.dfs(1:10)
> mapreduce(input = small.ints, map = function(k, v) cbind(v, v^2))
> from.dfs("/tmp/RtmpWnzxl4/file5deb791fcbd5")
```

因为 MapReduce 只能访问 HDFS 文件系统，所以先要用 to.dfs 把数据存储到 HDFS 文件系统中。MapReduce 的运算结果再用 from.dfs 函数从 HDFS 文件系统中取出。另外，在执行 from.dfs 之前系统会在屏幕中打印出 output 文件的位置。

rhdfs 包的使用

进入 R 环境之后，输入如下命令操作：

```
> library(rhdfs)

/* 成功启动后会提示你 hdfs.init() 需要开启，执行以下语句即可 */

> hdfs.init()
```

rhdfs 查看 Hadoop 目录

```
> hdfs.ls("/usr/")
```

查看 Hadoop 数据文件（在 hadoop 环境）

```
hadoop fs -cat /home/hadoop/input/fl.txt
```

在 RHadoop 实现的机器学习算法

RHadoop 并没有像其他算法库那样有开源的算法模型，但是它可以实现 Hadoop 的 map 和 reduce 操作。RHadoop 可以将 map 和 reduce 实现为 R 中的函数（function），从而实现分布式算法。所以，RHadoop 需要程序员自己开发算法代码。

RHadoop 的 KMeans 算法介绍：

首先来看一下 KMeans 的 RHadoop 实现代码，如图 5-91 所示。

从图 5-93 中可以看到 KMeans 的实现方式和 R 语言单机情形下是一样的，KMeans.mr 是一个函数（function），该函数实现 KMeans 算法。RHadoop 与单机版 R 最大的不同就是代码必须是分布式的，也就是说需要程序员编写 map 和 reduce 函数，如图 5-92 方框中所示。

由于是在 Hadoop 集群上运行，所以需要加载 rjava、rhdhfs、rmr2 这几个包。

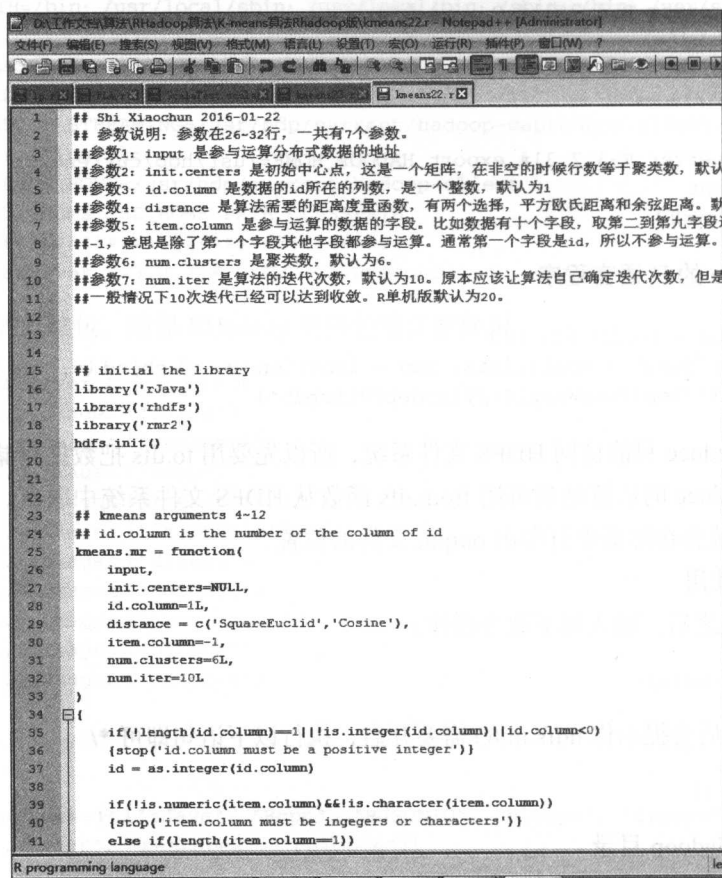


图 5-91 RHadoop 实现 KMeans

KMeans 算法运行结果：对已经编写完成的 RHadoop 代码，可以直接在 Hadoop 集群的 R 控制台中运行，如图 5-93 所示。

与调用单机版的 R 函数一样，只需要初始化并调用该函数，即实现了对数据的分布式算法。所有的计算结果都保存在变量 w（可以任意命名）之中。KMeans 算法的返回结果并不是非常大。这和 R 语言单机版的 KMeans 算法一样，如图 5-94 所示。

接下来解释调用 KMeans.mr 函数时的各参数配置：

```

## run the KMeans.mr function
w <- KMeans.mr (
  init.centers=NULL,
  input = '/user/root/ShiXiaochunInput/KMeansTestData02.txt',
  id.column=1L,
  distance = 'SquareEuclid',

```



```

item.column=-1L,
num.clusters=6L,
num.iter=6L
)

```

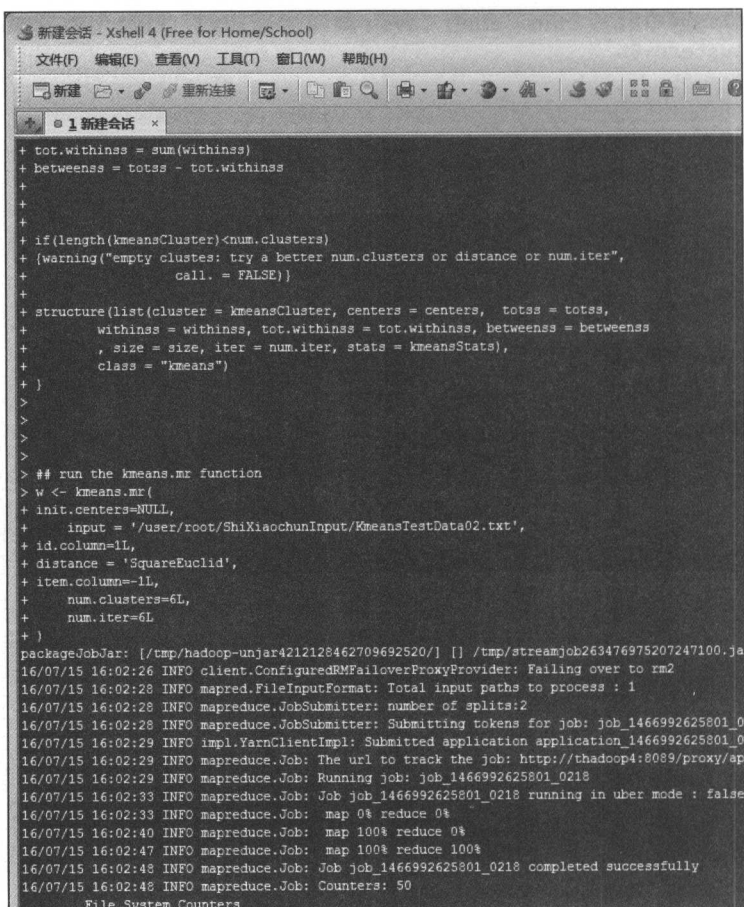
```

D:\工作文档\算法\RHadoop算法\K-means算法RHadoop版\kmeans22.r - Notepad++ [Administr
件(F) 编辑(E) 搜索(S) 视图(V) 格式(M) 语言(L) 设置(T) 宏(O) 运行(R) 插件(P) 窗口(W)
91 combine.in.memory = FALSE)
92 {
93
94     ## kmeansCenters.map
95     kmeansCenters.map =
96     function(., values) {
97         nearest = {
98             if(is.null(C))
99                 sample(
100                     1:num.clusters,
101                     nrow(values),
102                     replace = TRUE)
103             else {
104                 D = dist.fun(C, values[,item.column])
105                 nearest = max.col(-D)}
106
107             keyval(nearest, values[,item.column])}
108
109     ## kmeansCenters.reduce
110     kmeansCenters.reduce = function(k, values)
111     keyval(
112         k,
113         t(as.matrix(apply(values, 2, mean))))
114
115     ## kmeansCenters-main-1
116     C = init.centers
117     for(i in 1:num.iter) {
118         C =
119         values(
120             from.dfs(
121                 mapreduce(
122                     input = input,
123                     input.format = 'csv',
124                     map = kmeansCenters.map,
125                     reduce = kmeansCenters.reduce)))
126         ## kmeansCenters's mapreduce end
127     }
128     return(C)
129 }
130
131 ## run the kmeansCenters.mr

```

图 5-92 RHadoop 分布式 KMeans 改写

- ❑ 参数 `init.centers`：这是初始中心点，在不清楚的情况下一律设置为 `NULL`。
- ❑ 参数 `input`：这是分布式数据的存储地址，存储方式为 `hdfs`，支持 `txt` 与 `csv` 格式的数据。
- ❑ 参数 `id.column`：这是数据的 `id` 所在的列，R 语言计数从 1 开始，不像 `python` 从 0 开始。每一个数据都必然要有 `id`。



```

新建会话 - Xshell 4 (Free for Home/School)
文件(F) 编辑(E) 查看(V) 工具(T) 窗口(W) 帮助(H)

新建 重新连接

+ tot.withinss = sum(withinss)
+ betweenss = totss - tot.withinss
+
+
+ if(length(kmeansCluster)<num.clusters)
+ {warning("empty clusters: try a better num.clusters or distance or num.iter",
+   call. = FALSE)}
+
+ structure(list(cluster = kmeansCluster, centers = centers, totss = totss,
+   withinss = withinss, tot.withinss = tot.withinss, betweenss = betweenss
+   , size = size, iter = num.iter, stats = kmeansStats),
+   class = "kmeans")
+
+
+
+ ## run the kmeans.mr function
+ > w <- kmeans.mr(
+   + init.centers=NULL,
+   + input = '/user/root/ShiXiaochunInput/KmeansTestData02.txt',
+   + id.column=1L,
+   + distance = 'SquareEuclid',
+   + item.column=-1L,
+   + num.clusters=6L,
+   + num.iter=6L
+ )
packageJobJar: [/tmp/hadoop-unjar4212128462709692520/] [] /tmp/streamjob263476975207247100.jar
16/07/15 16:02:26 INFO client.ConfiguredRMFailoverProxyProvider: Failing over to rm2
16/07/15 16:02:28 INFO mapred.FileInputFormat: Total input paths to process : 1
16/07/15 16:02:28 INFO mapreduce.JobSubmitter: number of splits:2
16/07/15 16:02:28 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1466992625801_0
16/07/15 16:02:29 INFO impl.YarnClientImpl: Submitted application application_1466992625801_0
16/07/15 16:02:29 INFO mapreduce.Job: The url to track the job: http://thadoop4:8089/proxy/ap
16/07/15 16:02:29 INFO mapreduce.Job: Running job: job_1466992625801_0218
16/07/15 16:02:33 INFO mapreduce.Job: Job job_1466992625801_0218 running in uber mode : false
16/07/15 16:02:33 INFO mapreduce.Job: map 0% reduce 0%
16/07/15 16:02:40 INFO mapreduce.Job: map 100% reduce 0%
16/07/15 16:02:47 INFO mapreduce.Job: map 100% reduce 100%
16/07/15 16:02:48 INFO mapreduce.Job: Job job_1466992625801_0218 completed successfully
16/07/15 16:02:48 INFO mapreduce.Job: Counters: 50
File System Counters

```

图 5-93 RHadoop 分布式 KMeans 运算结果

- ❑ 参数 `item.column`：这是参数 `KMeans.mr` 函数计算的字段，也就是参与计算的列。对每一行数据来说，并不是所有的列都需要参与计算。特别地，`id` 所在的列是不能参与计算的，所以在这里 `item.column = -1L` 就是去掉第 1 列（与 `id.column = 1L` 对应），这和单机版的 R 一样。
- ❑ 参数 `num.clusters`：这是 KMeans 算法聚类的类数，在此是将数据聚为 6 类。
- ❑ 参数 `num.iter`：这是 KMeans 算法需要的迭代次数，每一次迭代都是一个 mapreduce 任务。

2. Mahout

Apache Mahout 是 Apache Software Foundation (ASF) 开发的一个全新的开源项目，其主要目标是创建一些可伸缩的机器学习算法，供开发人员在 Apache 许可下免费使用。该项目已经发展到了它的最第二个年头，目前只有一个公共发行版。Mahout 包含许多实现，包括集群、分类、CP 和进化程序。此外，通过使用 Apache Hadoop 库，Mahout 可以有效地扩展到

表 5-17 Mahout 算法汇总

算法类	算法名	中文名
分类算法	Logistic Regression	逻辑回归
	Bayesian	贝叶斯
	SVM	支持向量机
	Perceptron	感知器算法
	Neural Network	神经网络
	Random Forests	随机森林
	Restricted Boltzmann Machines	有限波尔兹曼机
聚类算法	Canopy Clustering	Canopy 聚类
	K-means Clustering	K 均值算法
	Fuzzy K-means	模糊 K 均值
	Expectation Maximization	EM 聚类（期望最大化聚类）
	Mean Shift Clustering	均值漂移聚类
	Hierarchical Clustering	层次聚类
	Dirichlet Process Clustering	狄里克雷过程聚类
	Latent Dirichlet Allocation	LDA 聚类
	Spectral Clustering	谱聚类
关联规则挖掘	Parallel FP Growth Algorithm	并行 FP Growth 算法
回归	Locally Weighted Linear Regression	局部加权线性回归
降维 / 维约简	Singular Value Decomposition	奇异值分解
	Principal Components Analysis	主成分分析
	Independent Component Analysis	独立成分分析
	Gaussian Discriminative Analysis	高斯判别分析
进化算法	并行化了 Watchmaker 框架	
推荐 / 协同过滤	Non-distributed recommenders	Taste (UserCF, ItemCF, SlopeOne)
	Distributed Recommenders	ItemCF
向量相似度计算	RowSimilarityJob	计算列间相似度
	VectorDistanceJob	计算向量间距离
非 MapReduce 算法	Hidden Markov Models	隐马尔可夫模型
集合方法扩展	Collections	扩展了 Java 的 Collections 类

Mahout 推荐算法介绍

我们先看一下 org.apache.mahout.cf.taste，如图 5-95 所示。

packages 的说明：

- ❑ common：公共类包括：异常、数据刷新接口、权重常量。
- ❑ eval：定义构造器接口，类似于工厂模式（什么是工厂模式请参考：<http://blog.chinaunix.net/uid-25958655-id-4243289.html>）。
- ❑ model：定义数据类型接口。
- ❑ neighborhood：定义近邻算法的接口。
- ❑ recommender：定义推荐算法的接口。
- ❑ similarity：定义相似度算法接口。

- transforms: 定义数据转换接口。
- hadoop: 基于 Hadoop 的分布式算法的实现类。
- impl: 单机内存算法实现类。

从上面的 packages 情况来看, 可以粗略地看出推荐引擎分为 5 个主要部分: 数据模型、相似度算法、近邻算法、推荐算法、算法评分器。从数据处理能力上看, 算法可以分为: 单机内存算法和基于 Hadoop 的分布式算法。

Taste 接口相关介绍

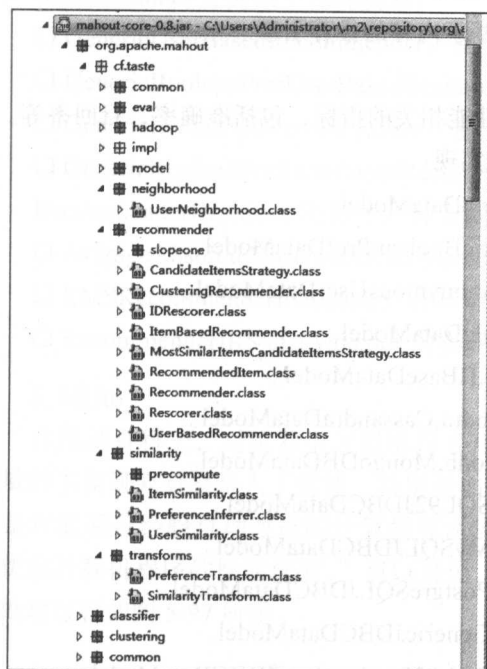


图 5-95 Mahout 推荐算法代码包

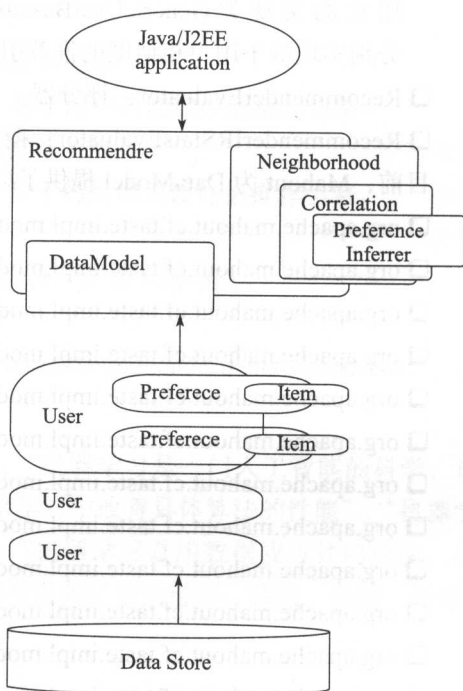


图 5-96 Mahout Taste 算法实现逻辑

Mahout 使用了 Taste 来提交协同过滤算法的实现, 它是一个基于 Java 实现的可扩展的、高效的推荐系统, Taste 既实现了最基本的基于用户的和基于内容的推荐算法, 同时也提供了扩展接口, 使用户可以方便地定义和实现自己的推荐算法。同时, Taste 不仅仅只适用于 Java 应用程序, 它可以作为内部服务器的一个组件以 HTTP 和 Web Service 的形式向外界提供推荐的逻辑, Taste 的设计使它能满足企业对推荐引擎在性能、灵活性和可扩展性等方面的要求。

Taste 主要包括以下几个接口:

- DataModel 是用户喜好信息的抽象接口, 它的具体实现支持从任意类型的数据源抽取用户喜好信息。Taste 默认提供 JDBCDataModel 和 FileDataModel, 分别支持从数据库和文件中读取用户的喜好信息。
- UserSimilarity 和 ItemSimilarity。UserSimilarity 用于定义两个用户间的相似度, 它是基

于协同过滤的推荐引擎的核心部分，可以用来计算用户的“邻居”，这里我们将与当前用户口味相似的用户称为他的邻居。ItemSimilarity 类似，用于计算内容之间的相似度。

❑ UserNeighborhood 用于基于用户相似度的推荐方法中，推荐的内容是基于找到与当前用户喜好相似的邻居用户的方式产生的。UserNeighborhood 定义了确定邻居用户的方法，具体实现一般是基于 UserSimilarity 计算得到的。

❑ Recommender 是推荐引擎的抽象接口，Taste 中的核心组件。程序中，为 Recommender 提供一个 DataModel，它就可以计算出对不同用户的推荐内容。实际应用中，主要使用它的实现类 GenericUserBasedRecommender 或者 GenericItemBasedRecommender，分别实现基于用户相似度的推荐引擎或者基于内容的推荐引擎。

❑ RecommenderEvaluator：评分器。

❑ RecommenderIRStatsEvaluator：搜集推荐性能相关的指标，包括准确率、召回率等。

目前，Mahout 为 DataModel 提供了以下几种实现：

❑ org.apache.mahout.cf.taste.impl.model.GenericDataModel。

❑ org.apache.mahout.cf.taste.impl.model.GenericBooleanPrefDataModel。

❑ org.apache.mahout.cf.taste.impl.model.PlusAnonymousUserDataModel。

❑ org.apache.mahout.cf.taste.impl.model.file.FileDataModel。

❑ org.apache.mahout.cf.taste.impl.model.hbase.HBaseDataModel。

❑ org.apache.mahout.cf.taste.impl.model.cassandra.CassandraDataModel。

❑ org.apache.mahout.cf.taste.impl.model.mongodb.MongoDBDataModel。

❑ org.apache.mahout.cf.taste.impl.model.jdbc.SQL92JDBCDataModel。

❑ org.apache.mahout.cf.taste.impl.model.jdbc.MySQLJDBCDataModel。

❑ org.apache.mahout.cf.taste.impl.model.jdbc.PostgreSQLJDBCDataModel。

❑ org.apache.mahout.cf.taste.impl.model.jdbc.GenericJDBCDataModel。

❑ org.apache.mahout.cf.taste.impl.model.jdbc.SQL92BooleanPrefJDBCDataModel。

❑ org.apache.mahout.cf.taste.impl.model.jdbc.MySQLBooleanPrefJDBCDataModel。

❑ org.apache.mahout.cf.taste.impl.model.jdbc.PostgreBooleanPrefSQLJDBCDataModel。

❑ org.apache.mahout.cf.taste.impl.model.jdbc.ReloadFromJDBCDataModel。

从类名上就可以大概猜出每个 DataModel 的用途，奇怪的是竟然没有 HDFS 的 DataModel，有人实现了一个，请参考 MAHOUT-1579 (<https://issues.apache.org/jira/browse/MAHOUT-1579>)。

UserSimilarity 和 ItemSimilarity 相似度实现有以下几种：

❑ CityBlockSimilarity：基于 Manhattan 距离相似度。

❑ EuclideanDistanceSimilarity：基于欧几里得距离计算相似度。

❑ LogLikelihoodSimilarity：基于对数似然比的相似度。

❑ PearsonCorrelationSimilarity：基于皮尔逊相关系数计算相似度。

❑ SpearmanCorrelationSimilarity：基于皮尔斯曼相关系数相似度。

□ TanimotoCoefficientSimilarity: 基于谷本系数计算相似度。

□ UncenteredCosineSimilarity: 计算 Cosine 相似度。

以上相似度的说明, 请参考 Mahout 推荐引擎介绍。

UserNeighborhood 的主要实现有两种:

□ NearestNUserNeighborhood: 对每个用户取固定数量 N 个最近邻居。

□ ThresholdUserNeighborhood: 对每个用户基于一定的限制, 取落在相似度限制以内的所有用户为邻居。

Recommender 分为以下几种实现:

□ GenericUserBasedRecommender: 基于用户的推荐引擎。

□ GenericBooleanPrefUserBasedRecommender: 基于用户的无偏好值推荐引擎。

□ GenericItemBasedRecommender: 基于物品的推荐引擎。

□ GenericBooleanPrefItemBasedRecommender: 基于物品的无偏好值推荐引擎。

RecommenderEvaluator 有以下几种实现:

□ AverageAbsoluteDifferenceRecommenderEvaluator: 计算平均差值。

□ RMSRecommenderEvaluator: 计算均方根差。

□ RecommenderIRStatsEvaluator 的实现类是 GenericRecommenderIRStatsEvaluator。

3. MLlib

在维基百科上, 对机器学习提出以下几种定义: “机器学习是一门人工智能的科学, 该领域的主要研究对象是人工智能, 特别是如何在经验学习中改善具体算法的性能”。“机器学习是对能通过经验自动改进的计算机算法的研究”。“机器学习是用数据或以往的经验, 以此优化计算机程序的性能标准。”可以看出, 机器学习强调三个关键词: 算法、经验、性能, 其处理过程如图 5-97 所示。

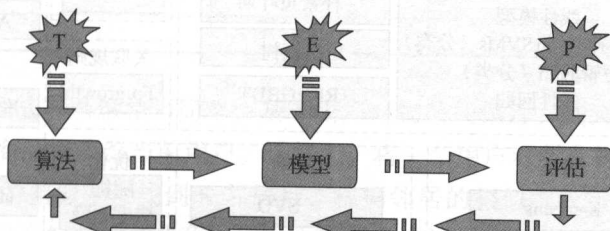


图 5-97 机器学习通用逻辑

图 5-97 表明机器学习是数据通过算法构建出模型并对模型进行评估, 评估的性能如果达到要求就用这个模型来测试其他的数据, 如果达不到要求就要调整算法来重新建立模型, 再次进行评估, 如此循环往复, 最终获得满意的经验来处理其他的数据。

Spark 作为新兴的、应用范围最为广泛的大数据处理开源框架引起了广泛的关注, 它吸引了大量程序设计和开发人员进行相关内容的学习与开发, 其中 MLlib 是 Spark 框架使用的

核心。Spark 之所以在机器学习方面具有得天独厚的优势，有以下几点原因：

机器学习算法一般都有很多个步骤迭代计算的过程，机器学习的计算需要在多次迭代后获得足够小的误差或者足够收敛才会停止，迭代时如果使用 Hadoop 的 MapReduce 计算框架，每次计算都要读/写磁盘以及任务的启动等工作，这会导致非常大的 I/O 和 CPU 消耗。而 Spark 基于内存的计算模型天生就擅长迭代计算，多个步骤计算直接在内存中完成，只有在必要时才会操作磁盘和网络，所以说 Spark 正是机器学习的理想平台。

从通信的角度讲，如果使用 Hadoop 的 MapReduce 计算框架，ResourceManager 和 NodeManager 之间由于是通过 heartbeat 的方式来进行通信和传递数据，因此会导致执行速度非常慢，而 Spark 具有出色而高效的 Akka 和 Netty 通信系统，通信效率极高。

MLlib (Machine Learning lib) 是 Spark 对常用的机器学习算法的实现库，同时包括相关的测试和数据生成器。Spark 的设计初衷就是为了支持一些迭代的 Job，这正好符合很多机器学习算法的特点。在 Spark 官方首页中展示了 Logistic Regression 算法在 Spark 和 Hadoop 中运行的性能比较，如图 5-98 所示。

可以看出，在 Logistic Regression 的运算场景下，Spark 比 Hadoop 快了 100 倍以上。

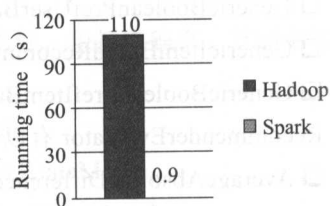


图 5-98 Logistic Regression 在 Spark 和 Hadoop 中运行时间对比

MLlib 算法介绍

主要的机器学习算法目前在 MLlib 中都已经提供了，例如分类回归、聚类、关联规则、推荐、降维、优化、特征抽取筛选、用于特征预处理的数理统计方法，以及算法的评测。支持的算法图谱如图 5-99 所示。

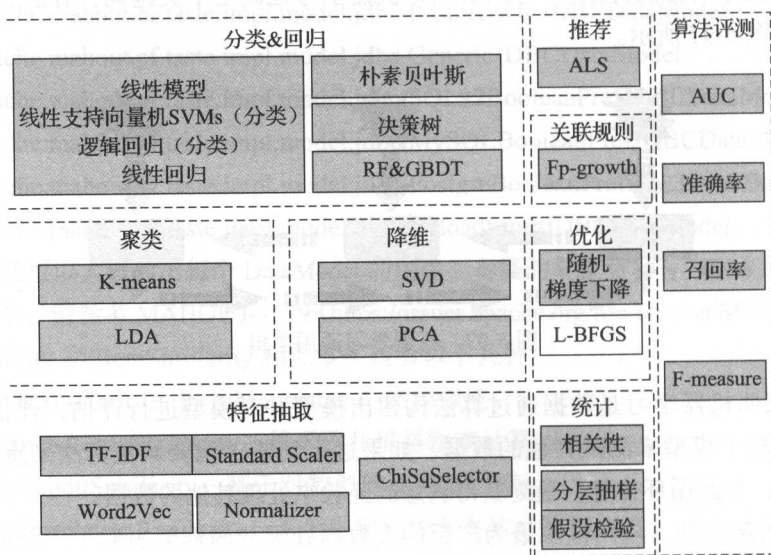


图 5-99 MLlib 支持算法汇总

案例介绍

我们以协同过滤算法为例介绍如何使用 MLlib 库来进行机器学习应用的构建。首先介绍什么是协同过滤算法。通常，协同过滤算法按照数据使用，可以分为：

□ 基于用户 (UserCF)。

□ 基于商品 (ItemCF)。


□ 基于模型 (ModelCF)。

按照模型，可以分为：

□ 最近邻模型：基于距离的协同过滤算法。

□ Latent Factor Mode (SVD)：基于矩阵分解的模型。

□ Graph：图模型、社会网络图模型。

 **提示** 本书使用的协同过滤算法是基于矩阵分解的模型。

(1) 基于用户 (UserCF) ——基于用户相似性

基于用户的协同过滤，通过不同用户对物品的评分来评测用户之间的相似性，基于用户之间的相似性做出推荐。简单来讲，就是向用户推荐和他兴趣相似的其他用户喜欢的物品。

如表 5-18 所示，有三个用户 A、B、C，四个物品 A、B、C、D，需要向用户 A 推荐物品。这里，由于用户 A 和用户 C 都买过物品 A 和物品 C，所以，我们认为用户 A 和用户 C 非常相似，同时，用户 C 又买过物品 D，那么就需要给 A 用户推荐物品 D。

表 5-18 基于用户相似性的推荐

用户 / 物品	物品 A	物品 B	物品 C	物品 D
用户 A	√		√	推荐
用户 B		√		
用户 C	√		√	√

基于 UserCF 的基本思想相当简单，基于用户对物品的偏好，找到相邻邻居用户，然后将邻居用户喜欢的商品推荐给当前用户。计算上，将一个用户对所有物品的偏好作为一个向量来计算用户之间的相似度，找到 K 邻居后，根据邻居的相似度权重以及他们对物品的偏好，预测当前用户没有偏好的未涉及物品，计算得到一个排序的物品列表作为推荐。

(2) 基于商品 (ItemCF) ——基于商品相似性

基于商品的协同过滤，就是通过用户对不同 item 的评分来评测 item 之间的相似性，基于 item 之间的相似性做出推荐。简单来讲，就是向用户推荐和他之前喜欢的物品相似的物品。

如表 5-19 所示，有三个用户 A、B、C 和三件物品 A、B、C，需要向用户 C 推荐物品。这里，由于用户 A 买过物品 A 和 C，用户 B 买过物品 A、B、C，用户 C 买过物品 A，从用

户 A 和 B 可以看出，这两个用户都买过物品 A 和 C，说明物品 A 和 C 非常相似，同时，用户 C 又买过物品 A，所以，将物品 C 推荐给用户 C。

表 5-19 基于商品相似性的推荐

用户 / 物品	物品 A	物品 B	物品 C
用户 A	✓		✓
用户 B	✓	✓	✓
用户 C	✓		推荐

基于 ItemCF 的原理和基于 UserCF 类似，只是在计算邻居时采用物品本身，而不是从用户的角度，即基于用户对物品的偏好找到相似的物品，然后根据用户的历史偏好，推荐相似的物品给他。

从计算角度，即将所有用户对某个物品的偏好作为一个向量来计算物品之间的相似度，得到物品的相似物品后，根据用户历史的偏好预测当前用户还没有表示偏好的物品，计算得到一个排序的物品列表作为推荐。

(3) 基于模型 (ModelCF)

基于模型的协同过滤推荐就是基于样本的用户喜好信息，训练一个推荐模型，然后根据实时的用户喜好的信息进行预测，计算推荐。本书使用的是基于矩阵分解的模型，算法如图所示。

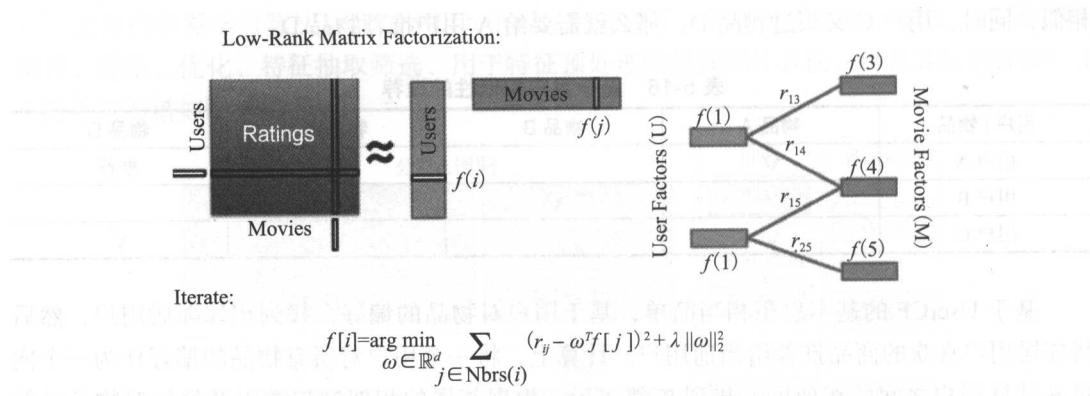


图 5-100 基于模型的推荐

Spark MLlib 当前支持基于模型的协同过滤，其中用户和商品通过一小组隐性因子进行表达，并且这些因子也用于预测缺失的元素。MLlib 使用交替最小二乘法 (ALS) 来学习这些隐性因子。

数据介绍

下面我们结合 MovieLens 公布的 10M 数据来动手训练一个协同过滤评分预测模型，数据样例如下。

打分数据片段, 字段含义依次为用户 ID:: 电影 ID:: 电影评分 :: 时间戳, 如图 5-101 所示。

用户数据片段, 字段含义依次为用户 ID:: 用户性别 :: 年龄 :: 职业 :: 邮编, 如图 5-102 所示。

	0	10	20
1	1::1193::5::978300760		
2	1::661::3::978302109		
3	1::914::3::978301968		
4	1::3408::4::978300275		
5	1::2355::5::978824291		
6	1::1197::3::978302268		
7	1::1287::5::978302039		
8	1::2804::5::978300719		
9	1::594::4::978302268		
10	1::919::4::978301368		
11	1::595::5::978824268		
12	1::938::4::978301752		
13	1::2398::4::978302281		
14	1::2918::4::978302124		
15	1::1035::5::978301753		
16	1::2791::4::978302188		
17	1::2687::3::978824268		
18	1::2018::4::978301777		
19	1::3105::5::978301713		
20	1::2797::4::978302039		
21	1::2321::3::978302205		
22	1::720::3::978300760		
23	1::1270::5::978300055		

图 5-101 推荐原始用户评分数据

1	F::1::10::48067
2	M::56::16::70072
3	M::25::15::55117
4	M::45::7::02460
5	M::25::20::55455
6	F::50::9::55117
7	M::35::1::06810
8	M::25::12::11413
9	M::25::17::61614
10	F::35::1::95370
11	F::25::1::04093
12	M::25::12::32793
13	M::45::1::93304
14	M::35::0::60126
15	M::25::7::22903
16	F::35::0::20670
17	M::50::1::95350
18	F::18::3::95825
19	M::1::10::48073
20	M::25::14::55113
21	M::18::16::99353
22	M::18::15::53706
23	M::35::0::90049
24	F::25::7::10023
25	M::18::4::01609

图 5-102 推荐原始用户信息数据

影片信息数据, 字段含义依次为影片 ID:: 影片名称 :: 影片类型, 如图 5-103 所示。

1	Toy Story (1995)::Animation Children's Comedy
2	Jumanji (1995)::Adventure Children's Fantasy
3	Grumpier Old Men (1995)::Comedy Romance
4	Waiting to Exhale (1995)::Comedy Drama
5	Father of the Bride Part II (1995)::Comedy
6	Heat (1995)::Action Crime Thriller
7	Sabrina (1995)::Comedy Romance
8	Tom and Huck (1995)::Adventure Children's
9	Sudden Death (1995)::Action
10	GoldenEye (1995)::Action Adventure Thriller
11	American President, The (1995)::Comedy Drama Romance
12	Dracula: Dead and Loving It (1995)::Comedy Horror
13	Balto (1995)::Animation Children's
14	Nixon (1995)::Drama
15	Cutthroat Island (1995)::Action Adventure Romance
16	Casino (1995)::Drama Thriller
17	Sense and Sensibility (1995)::Drama Romance
18	Four Rooms (1995)::Thriller
19	Ace Ventura: When Nature Calls (1995)::Comedy
20	Money Train (1995)::Action
21	Get Shorty (1995)::Action Comedy Drama

图 5-103 推荐原始影片数据

代码实现

首先对数据进行预处理, 将 MovieLens 数据转换成 Spark MLlib 中的 Rating 对象, 代码清单如图 5-104 所示。

数据转换完成后将数据通过抽样分成训练集和测试集两部分, 代码清单如图 5-105 所示。

```
def transferFunction(line: String): Rating = {
  val array = line.split(":")
  Rating(array(0).toInt, array(1).toInt, array(2).toDouble)
}
```

```
val data = CFRecommendTools.getData(dataPath, transferFunction, sparkContext)
val splitData = data.randomSplit(Array(0.8, 0.2))
val trainData = splitData(0)
val testData = splitData(1)
```

图 5-104 数据预处理

图 5-105 数据分区

接下来将转换完成的训练数据通过 Spark MLlib 进行训练，代码清单如图 5-106 所示。

```
/**
 *
 * @param trainData 训练数据
 * @param rank 特征数,也就是用户特征向量的长度
 * @param iterNum 迭代次数
 * @param lambda 正则化系数,建议为0.01
 * @param block 并行计算分区数,-1为自动选择
 */
def trainModel(trainData: RDD[Rating], rank: Int, iterNum: Int = 10, lambda: Double = 0.01, block: Int = (-1)) {
    model = ALS.train(trainData, rank, iterNum, lambda, block)
}
```

图 5-106 模型训练

有了模型以后便可以对测试数据进行预测，代码清单如图 5-107 所示。

```
/**
 * 评分预测
 * @param userItems
 * @return
 */
def predict(userItems: RDD[(Int, Int)]): RDD[Rating] = getModel.predict(userItems)
```

图 5-107 数据预测

最后通过对测试集的预测来计算模型的预测偏差，代码清单如图 5-108 所示。

```
/**
 * 误差评估,针对模型对实际数据进行预测,计算预测偏差
 * @param testData
 * @return
 */
def mse(testData: RDD[Rating]): Double = {
    val transferTestData = transferData(testData)
    val originData = testData.map(rating => ((rating.user, rating.product), rating.rating))
    predict(transferTestData).foreach(println _)
    predict(transferTestData).map(rating => ((rating.user, rating.product), rating.rating)).join(originData)
    .map(f => {
        val err = f._2._1 - f._2._2
        err * err
    }).mean()
}
```

图 5-108 数据结果评估

4. Python

Python 本身的数据分析功能不强，需要安装一些第三方扩展库来增强它的能力。常用到的库有 Numpy（数组功能）、Scipy（科学计算）、Matplotlib（输出图表）、Pandas（最强大的数据处理库）、StatsModels（统计建模库）、Scikit-Learn（强大的机器学习库）、Theano（机器学习库）、Keras（神经网络和深度学习的唯一选择）、Gensim（自然语言分析）、OpenCV（图像识别库），下面将对这些库的功能、优点进行介绍。

如果安装的是 Anaconda 版 Python, 那么它已经自带了以下库: Numpy、Scipy、Matplotlib、Pandas 和 Scikit-Learn, 如果安装的是标准版 Python, 则需要依次安装以上库以及相关的依赖库, 所以强烈建议直接安装 Anaconda。

用 Python 进行科学计算是很丰富的学问, 所涉及的一些库如表 5-20 所示。

表 5-20 Python 常用扩展库

扩展库	简介
Numpy	提供数组支持, 以及相应的高效的处理函数
Scipy	提供矩阵支持, 以及矩阵相关的数值计算模块
Matplotlib	强大的数据可视化工具、作图库
Pandas	强大、灵活的数据分析和探索工具
StatsModels	统计建模和计量经济学, 包括描述统计、统计模型估计和推断
Scikit-Learn	支持回归、分类、聚类等的强大的机器学习库
Theano	定义、优化、评估涉及多维数组的数学表达式
Keras	深度学习库, 用于建立神经网络以及深度学习模型
Gensim	用来做文本主题模型的库, 文本挖掘可能用到
OpenCV	实现图像处理和计算机视觉方面的很多通用算法

Numpy——数组库

Python 并没有提供数组功能。虽然列表可以完成基本的数组功能, 但它不是真正的数组, 而且在数据量较大时, 使用列表的速度就会慢得让人难以接受。为此, Numpy 提供了真正的数组功能, 以及对数据进行快速处理的函数。Numpy 还是很多更高级的扩展库的依赖库, 后面介绍的 Scipy、Matplotlib、Pandas 等库都依赖于它。值得强调的是, Numpy 内置函数处理数据的速度是 C 语言级别的, 因此在编写程序时, 应当尽量使用它们内置的函数, 避免出现效率瓶颈的现象 (尤其是涉及循环的问题)。

Scipy——科学计算库

如果说 Numpy 让 Python 有了 Matlab 的味道, 那么 Scipy 就让 Python 真正地成为半个 Matlab 了。Numpy 提供了多维数组功能, 但它只是一般的数组, 并不是矩阵。比如, 当两个数组相乘时, 只是对应元素相乘, 而不是矩阵乘法。Scipy 提供了真正的矩阵, 以及大量基于矩阵运算的对象与函数。

Scipy 包含的功能有最优化、线性代数、积分、插值、拟合、特殊函数、快速傅里叶变换、信号处理和图像处理、常微分方程求解和其他科学与工程中常用的计算, 显然, 这些功能都是挖掘与建模必备的。Scipy 依赖于 Numpy, 因此安装它之前得先安装 Numpy。安装 Scipy 的方式与安装 Numpy 的方法大同小异, 需要提及的是, 在 Ubuntu 下也可以用类似的

Matplotlib——制图库

不论是数据挖掘还是数学建模, 都免不了数据可视化的问题。对于 Python 来说, Matplotlib 是最著名的绘图库, 它主要用于二维绘图, 当然它也可以进行简单的三维绘图。它不但

提供了一整套和 Matlab 相似但更为丰富的命令，让我们可以非常快捷地用 Python 可视化数据，而且允许输出达到出版质量的多种图像格式。

Pandas——主力数据分析工具

Python 进行数据分析的主力工具——Pandas。Pandas 是 Python 下最强大的数据分析和探索工具（没有之一）。它包含高级的数据结构和精巧的工具，使得在 Python 中处理数据非常快速和简单。Pandas 构建在 Numpy 之上，它使得以 Numpy 为中心的应用很容易使用。Pandas 的名称来自于面板数据（Panel Data）和 Python 数据分析（Data Analysis），它最初被作为金融数据分析工具而开发出来，由 AQR Capital Management 公司于 2008 年 4 月开发出来，并于 2009 年年底开源。

Pandas 的功能非常强大，支持类似于 SQL 的数据增、删、查、改，支持 Excel 的筛选、排序、数据透视表等，并且带有丰富的数据处理函数；支持时间序列分析功能；支持灵活处理缺失数据等。

StatsModels——统计建模库

Pandas 着眼于数据的读取、处理和探索，而 StatsModels 则更加注重数据的统计建模分析，它使得 Python 有了 R 语言的味道。StatsModels 支持与 Pandas 进行数据交互，因此，它与 Pandas 结合，成为 Python 下强大的数据挖掘组合。

安装 StatsModels 相当简单，既可以通过 pip 安装，又可以通过源码安装。对于 Windows 用户来说，官网上甚至已经有编译好的 exe 文件以供下载。如果手动安装，需要自行解决好依赖问题，StatsModels 依赖于 Pandas（当然也依赖于 Pandas 所依赖的），同时还依赖于 pasty（一个描述统计的库）。

Scikit-Learn——主力机器学习库

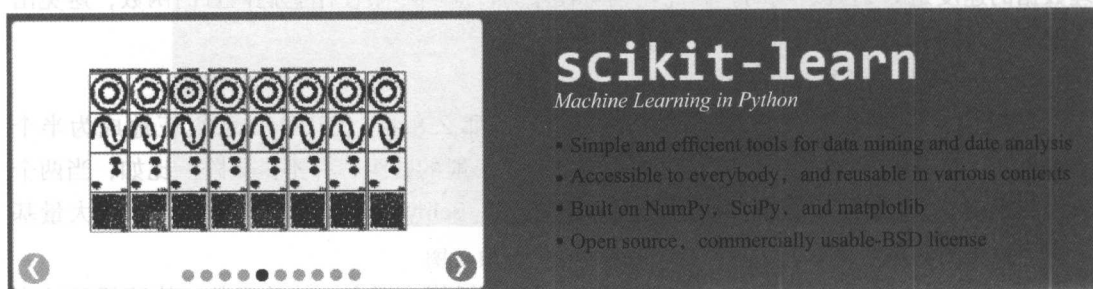


图 5-109 Scikit-Learn 机器学习库

Scikit-Learn 是 Python 下强大的机器学习工具包，包括数据预处理、有监督的分类算法、无监督聚类算法、降维算法、模型选择与评估指标等内容。在有监督的分类学习、聚类分析、算法评估方面功能非常强大，甚至可以做一些图像识别分析与语义分析。但是缺少预测方面的算法，以及神经网络和深度学习相关算法（需要使用后面介绍的 Keras 库）。

支持算法：

□ 数据预处理：数据标准化、数据正态化、异常值处理、缺失值填充、二值化、构建多项式特征、自定义转换。

□ 分类算法（有监督学习）：岭回归、逻辑回归、套索算法、贝叶斯回归、随机梯度下降、稳健回归、多变量线性回归、线性判别分析、二次判别分析、支持向量机、K 近邻算法、高斯过程、朴素贝叶斯、决策树、组合方法。图 5-110 是一个使用 Scikit-Learn 中所有的分类算法对一个样本数据进行分类的结果示例，从左到右依次的方法是：最近邻算法、线性支持向量机、RBF 支持向量机、决策树、随机森林、AdaBoost、朴素贝叶斯、LDA、QDA。

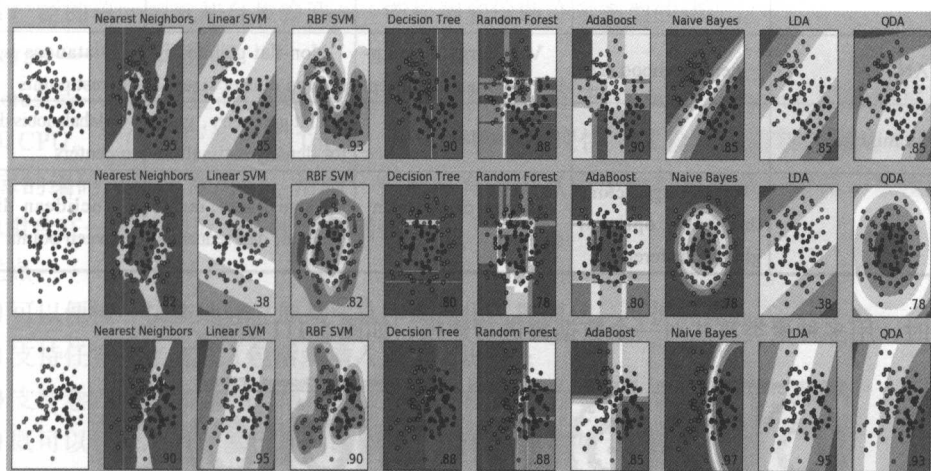


图 5-110 Scikit-Learn 算法汇总（分类部分）

□ 聚类算法：支持常规的 KMeans、Ward 等聚类算法，也支持最近 10 年新兴的 Affinity propagation 等聚类算法，如表 5-21 所示。

表 5-21 Scikit-Learn 聚类算法汇总

模型名称	参数	可测量性	用例	几何特征
KMeans	number of clusters	Very large n_samples, medium n_clusters with MiniBatch code		
General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points			
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Distances between points

(续)

模型名称	参数	可测量性	用例	几何特征
Spectral clustering	number of clusters	Medium n_samples, small n_clusters	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large n_samples, medium n_clusters	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large n_clusters and n_samples	Large dataset, outlier removal, data reduction.	Euclidean distance between points

同时，用多种聚类算法对一个数据集进行建模，如图 5-111 所示。

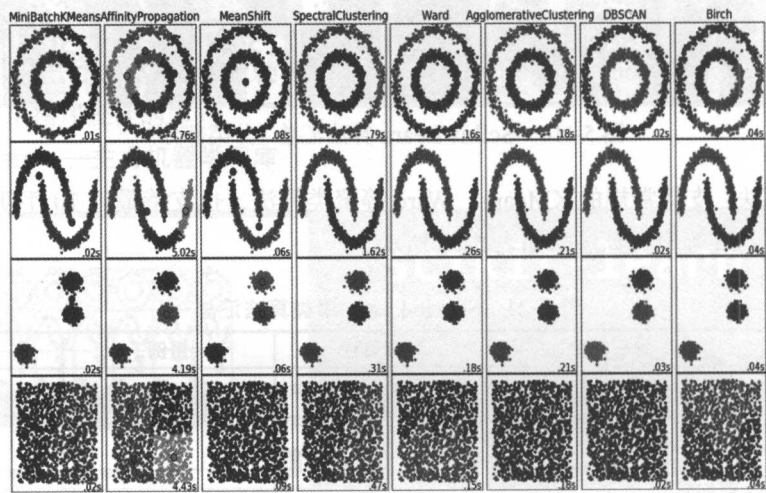


图 5-111 Scikit-Learn 聚类结果对比

□ 降维算法：包括主成分分析（PCA）、截断奇异值分解法、潜在语义分析、因子分析、独立成分分析、非负矩阵分解算法、文档主题生成模型（三层贝叶斯网络概率模型）。

Theano——机器学习库

Theano 是一个 Python 库，用来定义、优化、评估涉及多维数组的数学表达式。它提供了优良的数据结构来表示神经网络层，可以让用户轻松地编写 DeepLearning 模型，并提供运

行在 GPU 上的选择。与 Numpy 的数组类似，对线性代数来说十分高效。

特性：

- ❑ 紧密集成 Numpy——在 Theano 的编译函数中使用 `numpy.ndarray`；
- ❑ 高效的符号分解——Theano 替你计算一个或多个输入函数的推导；
- ❑ 速度和稳定性优化——即使 x 非常小也能正确得到 $\log(1+x)$ 的结果；
- ❑ 透明使用 GPU——使得执行数据密集型的计算速度高达 CPU 的 140 倍（仅对浮点数操作）；
- ❑ 动态生成 C 语言代码——计算表达式更快速；
- ❑ 广泛的单元测试和自我验证——能检测和诊断许多类型的错误。

Keras——神经网络库

Keras 是一个简约且高度模块化的神经网络库，基于 Theano 和 Python 语言编写，支持 GPU 和 CPU。它的设计参考了 Torch，开发侧重于实现快速试验和创造新的深度学习模型。Keras 库的编码风格非常简约、清晰。它把所有的要点用小类封装起来，这样一来便能够很容易地组合在一起并创造出一种全新的模型。

优点：

- ❑ 可以便捷、快速地建立原型（总体模块化、极简化的、可扩展化）；
- ❑ 支持任意连接方式（包括多输入多输出训练）；
- ❑ 支持卷积网络和递归网络，以及两者的组合；
- ❑ 既可以轻松地构建基于序列的网络，也可以创建基于图形的网络；
- ❑ 支持同时用 GPU、CPU 计算，使得计算密集型数据的速度提升数十倍。

算法演示：用 Keras 搭建神经网络模型的过程相当简洁，也相当直观，就像搭积木一般。通过短短几十行代码，我们就可以搭建起一个非常强大的神经网络模型，甚至是深度学习模型。简单搭建一个 MLP（多层感知器），如图 5-112 和图 5-113 所示。

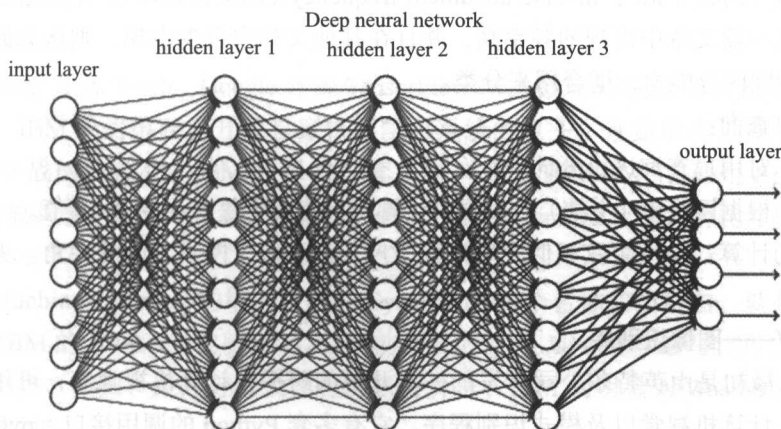


图 5-112 神经网络模型

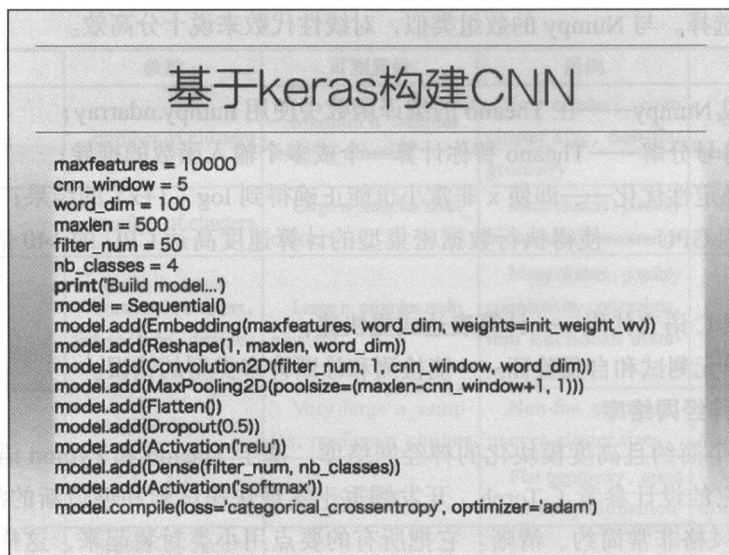


图 5-113 基于 Keras 的神经网络模型

Gensim——文本分类库

Gensim 是一个相当专业的文本主题模型 Python 工具包。在文本处理中，例如商品评论挖掘，有时需要了解每个评论分别和商品描述之间的相似度，以此衡量评论的客观性。评论和商品描述的相似度越高，说明评论的用语比较官方，不带太多感情色彩，比较注重描述商品的属性和特性，角度更客观。那么 Python 中有计算文本相似度的程序包吗？恭喜你，不仅有，而且很好很强大，那就是 Gensim。

文本相似度计算的需求始于搜索引擎。搜索引擎需要计算“用户查询”和爬下来的众多“网页”之间的相似度，从而把最相似的排在最前返回给用户。主要使用的算法是 tf-idf，tf：term frequency（词频），idf：inverse document frequency（倒文档频率）。主要思想是：如果某个词或短语在一篇文章中出现的频率高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。

处理用户查询：

第一步：对用户查询进行分词。

第二步：根据网页库（文档）的数据，计算用户查询中每个词的 tf-idf 值。

相似度的计算：使用余弦相似度来计算用户查询和每个网页之间的夹角。夹角越小，越相似。

OpenCV——图像识别库

OpenCV 最初是由英特尔公司开发的一套开源的跨平台计算机视觉库，可用于开发实时的图像处理、计算机视觉以及模式识别程序。它有多套 Python 的调用接口，pyOpenCV 库是 Python 下的 OpenCV 库，它不但很全面地对 OpenCV 的各种函数和类进行了封装，而且能在

OpenCV 的图像对象和 Numpy 数组之间进行互换。这样便同时扩展了 Numpy 的图像处理能力以及 OpenCV 的数组处理能力。

包含的算法：图像金字塔算法、Mean-Shift 算法、分水岭算法、SURF 特征提取算法等。用霍夫变换（Hough Transform）能够找出图像中的直线和圆。OpenCV 提供了如下 3 种霍夫变换相关的函数：

- ❑ HoughLines(): 检测图像中的直线。
- ❑ HoughLinesP(): 检测图像中的直线段。
- ❑ HoughCircles(): 检测图像中的圆。

下面的演示程序使用 HoughLinesP() 和 HoughCircles() 进行线段和圆的检测。它们的各种参数均可以在控制面板中进行调整。

分水岭（Watershed）算法的基本思想是将图像的梯度当做一个地形图。图像中变化小的区域相当于地形图中的山谷，而变化大的区域相当于山峰。从指定的几个初始区域同时开始向地形灌不同颜色的水，水面上升逐渐淹没山谷，并且范围逐渐扩大。当所有区域的水面连接到一起时，所得到的不同颜色的灌溉区域就是最终的图像分割结果，最终的分割区域数和初始区域数相同，如图 5-114 是 watershed() 的图像分割结果，从左往右逐步添加新的初始区域。



图 5-114 OpenCV- 图像识别库

5. SystemML

继 Facebook 开源 Torch、Google 开源 TensorFlow 以及微软开源分布式机器学习工具包 DMTK 之后，IBM 成为第四家开源自家机器学习系统的巨头，这显示出机器学习的生态构建与人才争夺战的白热化。IBM 开源的这套系统叫做 SystemML，它通过 Apache Software Foundation 开放共享，并允许开发者修改其代码，目前 SystemML 已作为孵化器项目被 Apache 接纳。

ML 是 Machine Learning 的缩写，所以 SystemML 这个名称相当直白，就是一套机器学习系统，由 IBM 的 Almaden 实验室近 10 年前开发。它用 Java 语言编写，可支持描述性分析、分类、聚类、回归、矩阵分解及生存分析等算法，IBM 的明星 AIWaston 就整合了不少 SystemML 的功能。

SystemML 是一个灵活的、可扩展的机器学习系统，SystemML 有以下几点显著的特征：

- SystemML 中的算法通常具有可定制性，这一点类似于 R 语言和 Python 语言。
- SystemML 引擎具有多种执行模式，包括 Spark MLContext 模式、Spark Batch 模式、Hadoop Batch 模式、Standalone 模式以及 JMLC 模式，基于不同的计算引擎模式可以适用广泛的应用场景。
- SystemML 具有自动的算法优化功能，可以基于数据和集群特性自动进行优化，从而确保效率和可伸缩性。

优势

如上所述 Google、Microsoft 和 IBM 分别发布并开源了自己的机器学习工具包 TensorFlow、DMTK 和 SystemML，下面就从不同的方面分析这三种机器学习工具包的特点。

(1) 功能特点

TensorFlow 是一个用来编写和执行机器学习算法的工具。计算在数据流图中完成，图中的节点进行数学运算，边界是在各个节点中交换的张量（Tensors——多维数组）。TensorFlow 负责在不同的设备、内核以及线程上异步地执行代码，目前支持 CNN、RNN 和 LSTM 等图像、语音和自然语言处理（NLP）领域最流行的深度神经网络模型。

DMTK 是一个分布式机器学习框架，它由参数服务器和客户端软件开发包（SDK）两部分构成。参数服务器支持存储混合数据结构模型、接受并聚合工作节点服务器的数据模型更新、控制模型同步逻辑；客户端软件开发包负责维护节点模型缓存（与全局模型服务器同步）、本地训练和模型通信之间的流水线控制以及片状调度大模型训练。它包含 DMTK 框架、LightLDA 和分布式词向量（Word Embedding）三个组件。

SystemML 则是一门灵活的、可伸缩的机器学习（ML）语言，支持描述性分析、分类、聚类、回归、矩阵分解以及生存分析等算法。DML 中指定的算法能够根据数据和集群特性使用基于规则和基于成本的优化技术动态地编译和优化。

(2) 执行环境与模式

TensorFlow 能够在台式机、服务器或者移动设备的 CPU 和 GPU 上运行，也可以使用 Docker 容器部署到云环境中。在处理图像识别、语音识别和语言翻译等任务时，TensorFlow 依赖于配备图像处理单元（GPU）的机器和被用于渲染游戏图像的芯片，它对这些芯片的依赖度比想象中的高。当前开源的版本能够运行在单机上，暂不支持集群。操作系统方面，TensorFlow 能够运行在 Linux 和 MacOS 上。

DMTK 采用了传统的客户端-服务器架构，有多个服务器实例运行在多台机器上负责维护全局模型参数，而训练例程（Routines）则使用客户端 API 访问并更新这些参数。为了适应不同的集群环境，DMTK 框架支持两种进程间的通信机制：MPI 和 ZMQ。应用程序端不需要修改任何代码就能够在这两种方式之间切换。DMTK 支持 Windows 和 Linux（测试环境为 Ubuntu 12.04）两种操作系统。

SystemML 有多种执行模式。在独立模式下，SystemML 能够运行在单台机器上，这样数据科学家就能够在本地开发算法，不需要分布式集群。另外，还可以将算法分发到 Hadoop

或者 Spark 上,从而利用已有的资源和技能。SystemML 能够运行于 Windows、Linux 以及 MacOS 上。

(3) 实现语言、API 接口及文档

TensorFlow 的核心是使用 C++ 编写的,有完整的 Python API 和 C++ 接口,同时还有一个基于 C 的客户端 API。目前 TensorFlow 的 Python API 需要 Python 2.7,对于 Python 3 的支持正在开发中。在 TensorFlow 的官网上 Google 提供了完整的 API 接口说明、白皮书与示例教程(包括针对初学者和专家的)。

DMTK 则是使用 C++ 编写的,提供了一个客户端 API 和 SDK。DMTK 的官网对 DMTK 框架、LightLDA、分布式词向量的应用场景、下载、安装、配置、运行以及性能等方面都做了详尽的介绍。

SystemML 使用 Java 语言编写,开发人员能够使用类似于 R 或者 Python 的语法表达算法,通过 Java、Scala 以及 Python 操作 SystemML。文档方面,SystemML 基本集中于 GitHub 上,包括构建、测试、独立模式运行命令以及一个线性回归的示例。

(4) 应用场景

Google 已将 TensorFlow 用于 Gmail (SmartReply)、搜索 (RankBrain)、图片 (生成图像分类模型——Inception Image Classification Model)、翻译器 (字符识别) 等产品。

虽然 Microsoft 并没有在 DMTK 的官网上透露其在内部的应用情况,但是从其功能特点以及定位来看或许更倾向于自然语言处理方面,例如文本分类与聚类、话题识别以及情感分析等。

SystemML 则是 IBM 研发了超过十年的机器学习技术,沃森 (Watson) 在几年前的大型活动中就整合了很多 SystemML 的机器学习功能,当然目前开源的 SystemML 依然在孵化阶段。

实现算法介绍

SystemML 作为机器学习工具包,实现了众多的机器学习算法,下面对 SystemML 的算法做一个简单的介绍。SystemML 的算法实现分为 6 个大类,分别为:

(1) 描述性统计 (Descriptive Statistics)

描述性统计分析,就是对一组数据的各种特征进行分析,以便于描述测量样本的各种特征及其所代表的总体的特征。描述性统计分析的项目很多,常用的如平均数、标准差、中位数、频数分布、正态或偏态程度等。

(2) 分类 (Classification)

分类是数据挖掘、机器学习和模式识别中一个重要的研究领域。目前 SystemML 实现了逻辑回归、支持向量机、朴素贝叶斯、决策树以及随机森林等分类算法。

(3) 聚类 (Clustering)

聚类分析又称群分析,它是研究(样品或指标)分类问题的一种统计分析方法,目前 SystemML 中仅仅实现了 KMeans 聚类算法。

(4) 回归 (Regression)

回归分析是确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。目前

SystemML 实现的回归包括：线性回归、逐步线性回归、广义线性模型、阶梯式广义线性回归、回归算法评估等模型。

（5）矩阵分解（Matrix Factorization）

矩阵分解是将矩阵拆解为数个矩阵的乘积，可分为三角分解、满秩分解、QR 分解、Jordan 分解和 SVD（奇异值）分解等。SystemML 中提供了主成分分析和交替最小化矩阵分解两种算法。

（6）生存分析（Survival Analysis）

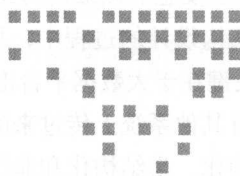
生存分析是指根据试验或调查得到的数据对事务的生存时间进行分析和推断。SystemML 中提供了 Kaplan-Meier 生存分析和 Cox 比例风险回归模型两种分析模型。

5.4 本章小结

通过 Hadoop 自身就能解决海量数据的所有问题吗？答案当然是否定的，对于大部分的企业来说，除了 Hadoop，必然会用到 NoSQL、流式计算、实时计算、全文索引等相关核心组件，以及内存数据库、消息中间件、数据可视化、数据缓存等相关组件。限于篇幅，本章仅列出了冰山一角。

本章需要读者重点掌握的知识点包括：

- ❑ 掌握大数据核心技术的所有内容，尤其是 Hadoop 基本内容、NoSQL 和实时计算，几乎每个企业都必不可少；
- ❑ 对于本章中非结构化数据的技术，例如全文检索、中文分词等根据企业需要进行学习和掌握；
- ❑ 对于相关技术中，根据不同的需要了解各自内容，重点是算法库部分。



大数据架构设计

随着企业级数据处理需求的不断发展，常规的数据架构往往难以满足需求。企业大数据架构是以数据管理平台（DMP）为基础，以不同的业务种类划分成多个数据区域（称之为“数据池”或“数据湖”）。它是用来存放企业所有类型数据的平台，其中包括企业通过原始积累、外部合作等，从自己内部、外部所有渠道获取到的结构化数据、半结构化数据和非结构化数据。该平台并不因为功能或技术上的原因显得独特，它本身会因存储着企业的所有内容而成为一个有着巨大运营意义的系统。另外，它可以通过大数据特有的分布式技术，使整个平台能够运行在非常廉价的商用机器上。

在企业大数据架构设计的过程当中，一般可通过许多方式来存储和分析。每个数据来源都有不同的特征，包括数据的频率、数据量、传输速度、数据类型和数据真实性。而处理大数据时会涉及更多维度，例如数据清洗、数据提升、安全性和存储策略。选择一种架构并构建合适的大数据解决方案极具挑战，因为需要考虑非常多的因素。

在本章中会通过之前的一些实战经验，阐述企业大数据架构设计的理念，而此架构设计可以实现企业系统级数据和大数据的完美整合，这些内容主要围绕架构设计原则、要素、模式进行。

6.1 大数据架构设计原则

大数据平台建设是一个长期的循序渐进的过程，也是一个不断创新和完善的过程，其伴随着企业 IT 系统的发展而不断完善。所以大数据架构设计的原则是：根据企业自身的业务特点为系统建设现状和未来发展蓝图，并依据数据类型对业务问题进行合理地分类，打造一个

可扩展、高可用、安全、高效的海量数据处理和挖掘的大数据平台。

在大数据架构设计的过程中，可能某些技术格外吸引眼球，例如 Hadoop、NoSQL 或者其他技术，但关键在于大数据平台汇集了企业的所有数据，所以此平台功能主要为：

- ❑ 收集所有其他系统上传过来的数据，同时进行规范化的分类存储。
- ❑ 支持结构化、半结构化和非结构化数据的实时上传、存储、分析和计算功能。
- ❑ 支持所有常用的数据模型及算法在此平台之上的运行计算。
- ❑ 提供通用的数据服务接口，使平台中的数据能够发送给其他系统使用。

除以上四点之外，围绕在大数据平台周围的还有一些补充和额外的技术，它们可以从那些存储在平台中的数据获取更多的认知和价值，共有五大类：数据可视化、高速缓存、消息系统、数据挖掘和 ETL。

- ❑ 数据可视化：简洁而美观的可视化报表技术可以使数据更直观地展现在大家的面前，它可以出色地执行报告任务，并且持续不断地为企业各个岗位的人员提供洞察功能，相信大部分企业都已经在不同程度上使用了一些 BI 工具。然而大数据平台需要提供更加复杂和专业的分析报告服务，开源的可视化技术如 H5、HighCharts、ECharts、D3 等都是大数据平台可视化的一些选择。

- ❑ 高速缓存：开源社区里关于高性能 Key-Value 数据库的发展非常迅速，该类数据库的好处是能够更加快速地读写即时数据，使即时产生的临时数据能够更加高效地进行交互。目前，使用最多的是 Redis 和 Memcached 等，在大数据平台中一般使用该类型的组件来做计算结果的缓存和高并发读取。

- ❑ 消息系统：稳定而高效的消息系统可以使企业中的各种消息数据进行稳定而安全的传输，该系统能够实时地收集反馈信息，并能支持大文件和高吞吐量的传输，使 TB 级的消息存储能够保持长时间的稳定运行。在企业实际应用场景中，“订单系统”所产生的数据可以通过消息系统进行转发，同时还可以在传输过程中进行一些简单的数据处理与聚合，目前比较强大的消息系统有 Kafka 和 MQ 等技术。

- ❑ 数据挖掘：这是数据科学家的专业领域，数据挖掘工具主要是通过统计规则、数学算法、数据关系等方法，在海量数据中挖掘出对特定场景有用的信息。Mathout、Python 和 R 语言都是此类中比较流行的工具。

- ❑ ETL：最后是大家所熟知的 ETL，它会协助我们进行各种操作、提取和处理，在大数据平台中，可实现 ETL 的技术和工具包括：Python、Scala、R 等，其中最重要且使用最多的还是 Hadoop 原生提供的 HiveSQL 和 MapReduce。

1. 架构设计的突破性原则

建设企业级大数据平台首先面临的挑战就是企业自身的数据结构，以及企业现有数据库的设计理念和原则。在大数据平台之前以关系数据库为核心的数据结构设计中，不少开发人员都有过专门为基于文档、图片进行数据存储的数据库表设计经历。但在大数据平台之上，

我们需要同时对多种不同格式的数据进行混合录入和存储，这就必须意识到曾经的原则已经不再适用，针对不同的数据结构定制不同的存储方案；同时需要全新的存储原则，即一种存储方案存储所有数据类型。

2. 数据存储的共存性原则

不少企业在大数据平台建设的过程当中，很容易有一个共同的误解：关系数据库全部的数据往 Hadoop 上导入后，就把原有的关系型数据库废弃了，到最后发现海量数据分析的性能问题解决了，但数据即时查询的效率问题却呈现出来了；或者仅仅将日志等非结构化数据写入 NoSQL 数据库中，由于关系型数据库和 NoSQL 无法进行有效的交互，因此无法进行有价值数据的关联分析。企业级大数据平台中的数据存储和存储之后相关的处理、分析挖掘和即时查询等功能，最成熟的存储架构应该还是基于 Hadoop 系列、关系型数据库、分析挖掘工具和查询分析语言等所有组件的全面搭配使用，而不是仅仅依赖某一两种。

所以企业大数据平台数据存储的共存性原则是：在 Hadoop 平台上存储包括了结构化和非结构化的原始数据，以及基于原始数据进行分析挖掘的“离线数据”，同时配合经过 Spark、Strom 进行流式和即时处理的“实时数据”；最后在关系型数据库和内存数据库中存储，为业务人员和在线系统提供结构化结果数据查询的“在线数据”。

3. 数据平台的实用性原则

企业级大数据平台为了能够提供更好的决策、更好的个性化和更好的交互性，在设计时应该要考虑到层级关系和高可用性，以便使其他应用系统也能够享受大数据平台所带来的高效、实时的价值输出点。值得注意的是，企业级应用系统都是按数据库系统来设计的，每秒钟最大限度只能处理百万计数据的任务事件，这与那些设计用来保存万亿计数据并生成复杂报告的大数据平台是截然不同的。如果能够通过大数据平台进行海量数据的计算，并使应用系统能够使用到计算结果，就可以使应用系统的运行效率、系统功能大幅度提升，那么企业级应用系统的价值随之也会大大提高。

6.2 大数据核心架构要素

我们的目标不仅仅只是建设一个高价值的大数据平台，而是让平台有更好的扩展性和开放性，使企业级应用系统共享大数据平台的能力，来帮助企业整个 IT 平台拥有一个质的飞跃。结合我们之前大数据平台设计和实施的项目经验，总结出大数据架构设计原则的基本要素如下：前瞻性、可扩展、开放性、高性能、稳定性、安全性、易维护、实用性、高可用、统一管理，如图 6-1 所示。



图 6-1 大数据核心架构要素

1. 前瞻性

建设一套成熟的大数据平台，首先要在技术和架构上具有一定的前瞻性。随着大数据平台技术不断成熟，就需要更多高级功能和特性。扩展到云端，并加入深度挖掘等新功能的能力迫切需要具有相应能力的平台。通过部署 Apache Hadoop 生态组件及 Spark、Mathout 等计算分析组件，对开发更深层次数据探索能力的大数据战略，并通过云平台服务优化现有平台功能的企业而言非常重要。从大数据平台前瞻性出发，需要考虑以下几个方面：

- 架构和平台统一：保持架构的统一和稳定，实现应用系统与服务接口的完全标准化，与企业所有业务系统和数据中心的协同统一，实现平台架构、资源和数据的共享。
- 技术先进：结合企业信息技术进行实际情况，即时跟随开源社区各组件的最新动态和技术优化，引入最稳定的前沿技术对已有技术进行不断改造升级，确保企业大数据平台的先进和实用。
- 安全实效：在平台设计过程中，应考虑现有软硬件的基础设施，以及原有业务系统的快速恢复能力，以实际需求为导向，对原有可复用的系统和模块的服务化进行改造，避免重复开发工作，并尽量复用原有硬件资源，节约投入成本。

2. 可扩展

可扩展性是指大数据平台在实施之后能够支持业务系统和应用系统发展的需要，可以动态扩展平台功能，并以服务接口的方式无缝对接其他应用系统。扩展性需要考虑以下几种场景：

- 服务器和计算节点扩展性：随着分发数据规模的扩大和推送节点的增多，对交换处理和传输处理的性能要求会越来越高，必须支持集群的方式进行扩展，所以需要大数据平台层可以扩充分布式集群的节点，提升存储和计算能力，平台中的每一种服务器都使用集群扩展模式，可以通过对服务器数量的增加获得更好的数据处理和查询能力。
- 数据结构扩展性：设计数据层和处理层模型时应充分考虑，除了能够容纳现有源系统的结构设计，还应该尽可能满足即将要上线的业务系统数据模型，同时还需要制定一套合理的模型设计规范，使得新上线的业务系统数据模型能很方便地扩展到大数据平台中。
- ETL 处理扩展性：ETL 扩展性有两个方面，增加新的 ETL 任务处理以及原有任务所处理的数据规模加大，ETL 处理架构必须能适应新的变化，需要考虑通过集群的方式来扩展。
- 平台接口扩展性：平台中的数据或算法模型不仅只针对平台本身的应用，还需要提供对外的数据交换服务接口，以便其他系统的数据或模型可以方便地集成进来，同时通过数据交换服务接口把数据或模型提供给其他应用系统进行使用。此外，还必须提供数据交换服务的二次开发接口，如图 6-2 所示。

3. 开放性

企业在做大数据技术选型，特别是 Hadoop 选型时，经常会受到一些供应商的“蛊惑”，

导致在开源版本和多家公司不同商业版本的选择上举棋不定，其实市场上大部分商业版本 Hadoop 都是基于开源版进行二次封装的。从平台整体的维护成本和学习成本两方面考虑，建议企业在初次建设时选择开源版本（Apache）或者更贴近开源版的商业版本（CDH），因为该版本的平台在整体设计和实现上，依托通用的大数据开源项目，遵循了业界广泛认可的事实标准，可以充分借力全球生态圈的资源，不但可以持续获取技术演进的红利，也可以拥有广泛的合作生态圈资源。

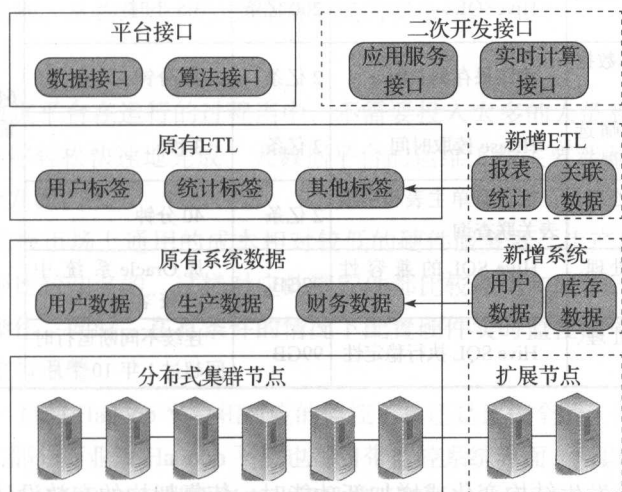


图 6-2 平台接口扩展结构

在硬件层面，建议选择相对知名和稳定的硬件设备。从平台的接口开放程度和自主二次开发的方面看，尽量不要使用所谓“硬软件一体化平台”。同时在架构上使软硬件分层解耦，可兼容多种异构物理设备，避免厂商绑定情况。最后在数据层面，支持多种数据源，包括结构化、非结构化类型的数据存储与处理，数据本身、数据计算也都需要提供接口支持开放共享。

4. 高性能

高性能是指在硬件资源有限的情况下，大数据应用开发平台及实施服务应尽可能地支持尽量多的数据服务需求，还能承受用户峰值时间段压力。在企业中对于大数据分析的需求日益成熟的今天，提供多个同步存储和分析数据的方法，使得大数据平台能够满足所有部门的性能需求。

大数据分析依赖于及时处理和查询复杂数据的能力，一个典型的大数据性能场景：需要建设一个数据仓库用来统计从网站日志中收集到的数据，分布式计算系统有能力在 15 分钟内处理 1 亿条以上的记录，而传统的关系型数据库则在相同时间内只能最高处理 100 万条记录。能否识别正确的基础设施来支持快速的数据可用性和高性能查询就意味着成功还是失败，以下是在实际应用中某企业级大数据平台的性能指标，如表 6-1 所示。

表 6-1 某企业大数据平台性能指标

序号	用例场景	测试接入点	数据大小	执行结果	集群信息
1	将 3 个月业务数据导入到 HDFS 文件系统	文件上传 HDFS	99GB	15 分钟	CPU：32 核，Memory：64GB，Disk：8TB，集群数量：10 台
2	将 HDFS 文件系统中 7 月份订单数据导入 Hive 文件系统	HDFS 加载到 Hive	99GB	5 秒左右	
3	200 亿条数据三表关联查询	HiveSQL	200 亿条	3.5 小时	
4	将计算后的目标数据导入 HBase 中	将结果存到 HBase	2 亿条	30 分钟	
5	通过查询条件筛选 HBase 中的数据并展示	HBase 读取时间	2 亿条	3 秒	
6	多场景对数据处理、计算	汇总订单主表和明细表关联查询	2 亿条	40 分钟	
7	多场景对数据处理、计算	Hive SQL 的兼容性测试	99GB	原 Oracle 系统中 SQL 兼容度 95%	
8	多场景对数据处理、计算	Hive SQL 执行稳定性	99GB	连续不间断运行时间超过 1 年 10 个月	

5. 稳定性

稳定性是指平台发生结构变化或增加新功能时，依靠架构的有效设计，仍然能保证正常运行。在大数据平台设计中，稳定性主要需要体现在以下几个方面：

- ❑ 数据模型的稳定性：模型的设计应能屏蔽数据源系统结构的变化发生时，对大数据平台和基于平台之上的应用系统带来的影响。局部数据模型的扩展不会对其他数据模型产生影响。
- ❑ 系统运行的稳定性：当正在运行的平台出现异常时，应具备快速实时的备份恢复机制，确保系统能及时恢复处理。当某个运行节点出现问题时，其他节点能够实现实时无缝切换。同时，各系统或功能模块在设计时应考虑自身的稳定运行策略。

6. 安全性

在大数据平台中，安全性主要包括两个层面的含义：一是防止数据服务体系的数据资源被恶意修改和盗取；二是防止数据在传输过程中被截留和篡改。所以安全性的设计具体体现在以下方面：

- ❑ 对于数据系统方面的安全性，主要依赖于各应用系统对用户角色和功能权限的控制。因此，在制定数据服务体系的应用系统设计开发规范时，应明确要求应用系统必须充分考虑安全性的设计。若已经建设了面向用户管理的统一用户认证平台，可以考虑通过统一用户认证平台来管理用户权限。
- ❑ 对于数据范围方面的安全性，在梳理出大数据应用开发平台及实施服务应用需求与目

标用户权限关系之后,通过在程序中对数据进行过滤,用户无法涉及其权限范围以外的数据,以确保数据范围的安全。数据过滤程序可抽象为一个准确、高效、易管理维护的过滤器。

- 对于数据传输的安全性,主要依赖于文件传输过程中的加解密处理。因此,在进行总体设计时需要充分考虑数据传输过程中的安全性。
- 此外,系统在进行网络规划时,对系统的安全级别也需要进行分析,必要时需要提高网络的安全级别,从物理设计层面提高系统的安全性。

7. 易维护

易维护是指大数据平台在运行的过程当中,不需要投入太多的人员和精力,使平台在出现故障或者升级时能够轻松快速地完成。大数据平台的运维工作主要涉及硬件设备、Hadoop 平台、应用系统三个方面:

- 硬件设备:选择市场上通用的成本相对较低的硬件服务器,生产厂商也尽量选联想、惠普、戴尔等比较知名的,这样对应的零配件都比较通用和廉价,能够快速替换和易于运维人员操作。同时,在有条件的情况下配置硬件实时监控的可视化平台,以便更好地及时监控与预警。
- Hadoop 平台:目前 Hadoop 平台相对应的监控工具还是比较全的,例如 Ganglia、Nagios 等。另外,大部分商业版 Hadoop 平台也都自带监控系统界面,所以在平台建设过程中,只需要选择合适的监控工具搭建起来,同时配一两位专业的 Hadoop 运维人员即可。
- 应用系统:应用系统运行过程中,最好及时收集完整准确的运行日志,并考虑开发简单易用的维护诊断工具,能帮助运维人员提供方便、可靠的维护操作手段,包括远程诊断和远程维护功能。

8. 实用性

在大数据平台的实用性方面,一定要避免为了跟风而使用某些新技术的情况,例如很多企业仅仅是需要海量数据的存储和离线分析、查询功能,并没有实时流量日志和用户行为,也没有其他太多的需求场景,在此情况下,像 Spark、Strom 等实现计算组件就完全没必要部署应用。值得注意的是,在集群中部署多套组件会使整个集群的运行效率降低,所以一定要从实际需求出发注意以下几点:

- 先从单一需求的最小规模的集群起步,然后线性扩展以满足不同的业务场景。按不同投资计划和规模的要求,可以仅从 5 台服务器资源开始。
- 从单一技术组件解决单一技术问题开始,根据技术需求不断扩展,切记贪大求全。
- 以“开源免费”为原则,前期以最小投入解决问题,当开源组件解决不了问题时再考虑收费组件。

9. 高可用

高可用是指尽可能避免重复投入,应尽可能考虑包括物理设备、系统软件、框架组件、

规范方法以及业务应用等多个层面的复用。在大数据平台建设过程中高可用的设计具体表现在以下几个方面：

- ETL 功能组件：在设计 ETL 任务处理流程时，要分析 ETL 任务的各个环节，尽可能找出一些公用的 ETL 组件，进行必要的封装，便于在模块内复用，进而推广到项目内进行复用。
- 数据预处理层的数据模型：在设计数据预处理层的数据模型时，应充分考虑应用系统的数据加工需求，尽可能将一些共性的加工需求在该层实现；并通过这种机制，不断扩充和完善改成的数据模型，实现加工数据的复用。
- 组件复用：各模块在开发的过程中，注意提炼出一些可用于共用的公共组件，在模块内实现复用，甚至在模块间实现复用。
- 硬件部署：在进行硬件部署的规划时，应充分对系统的处理规模进行分析。如果性能允许，尽可能集中部署，使用现有设备，在硬件方面实现复用。

10. 统一管理

基于大数据平台的应用系统设计时，建议设计相对应的平台管理功能，把硬件监控、Hadoop 平台监控都集成进来。建设企业级一体化的监控与管理平台，后期便于更好地统一管理和统一维护。统一管理平台应有功能如下：

- 关键功能的故障都有启停控制和恢复管理功能，包括集群节点管理、计算节点管理、进程管理、接口管理、服务管理和应用管理等。
- 通过管理任务流程的功能，避免单个节点或进程的运行错误导致整个业务全中断。
- 提供用户统一管理功能，实现平台中多个用户的资源隔离和任务优先级调度，避免不合操作规范的用户导致其他用户重要任务资源的运行受阻。

6.3 大数据架构设计模式

在设计和架构中，必须清晰地认识到没有万能的软件架构能解决所有问题，不同的场景、需求、限制下需要有针对性的架构模式才能满足大数据项目需求。结合之前的实际项目经验，我们总结出大数据架构设计模式需要从分层、分割、分布式、集群、缓存、异步、灾备、自动化几个方面考虑。

1. 分层

大数据平台从逻辑上通常分为数据源层、数据预处理和存储层、数据计算分析层和数据消费层。这种分层设计的原则：第一，可以保证各个层之间保持较好的解耦特征，使得各层的技术组件不会因为其他组件的版本更变、迭代更新等造成自身的应用障碍；第二，在技术开发时，有利于保障各层之间独立且并行开发，这种非线性的开发模式能有效加快平台实施和进度；第三，分层的设计模式能使各集群分别部署和开发，能够实现系统的扩展性并利于

维护。各层的功能具体如下：

- 数据源层：大数据平台内外部的结构化、非结构化的数据库、文本、文件等数据系统，第一方和第三方的集成数据提供商，内外部应用服务系统、内容管理系统、行为系统、监控系统、运营系统等。
- 数据预处理和存储层：经过数据抽取、集成、加载以及出于数据质量考虑而清洗过的数据，存储在大数据平台中，包括数据集市、数据仓库、分布式数据存储、分布式文件存储等。
- 数据计算分析层：所有有关数据计算、数据算法、机器学习、数据挖掘、实时计算、离线计算等部分都在这一层，满足上层数据消费所需要的各种实时和离线计算。
- 数据消费层：面向终端的程序、用户的产品、报表、分析、服务、接口等，以及与内外部应用系统的集成等。

2. 分割

分割是根据不同的业务主体，将整体业务体进行切割并细分到多个小业务，然后通过各自的集群来实现各自的业务应用。相对于侧重于流程性的纵向分层，分割更侧重于功能性的横向分层。

这种方式能够实现业务功能的独立开发，对某个业务模式或功能模块的修改不会过多地影响到其他业务模块的功能实现；同时，分割的架构设计方式还能在各个模块发生故障时，不影响其他模块的功能实现，防止整体性和串联型故障。图 6-3 所示为某智慧城市项目中业务分割示例。



图 6-3 业务分割模式

3. 分布式

分布式的架构设计是大数据系统的基础，它包括控制系统、接口系统、数据系统、应用系统等不同规范的分布式。它与集群的不同之处在于：分布式系统是由一个业务分拆多个子业务，将不同的业务分布在不同的地方，它是一种工作方式。而集群系统是将几台服务器集中在一起，实现同一业务，它是一种物理形态。在一个分布式系统中，多台服务器展现给用户的是一个统一的整体，就好像是一个系统。由于分布式架构相对较复杂，需要投入大量的开发和运维人员，所以在实施过程中并非越多越好。以下分别介绍几类分布式系统：

- 分布式控制系统：从大模型服务器、网络统一管理角度出发，按功能分散、管理集中的原则构思，采用多层分级、合作自治的结构形式，结合服务器、网络、虚拟化等

基础平台技术，可以对企业自身的基础 IT 设施进行统一升级改造。比如，目前主流的 OpenStack、KVM 等云计算组件都属于该类系统。

- ❑ 分布式接口系统：能运行在不同机器上，通过分布式接口就可以无须借助第三方软件或硬件进行数据交换和集成。所以分布式接口系统可以使企业内部不同平台、编程语言和组件模型中的不同类型系统进行数据交互和沟通。分布式接口组件有：WebService、RESTful API 等。
- ❑ 分布式数据系统：在传统数据库或单机版数据缓存组件不足以支撑企业不断增长的数据量的情况下，分布式数据系统可以解决该类问题。典型的分布式数据系统包括：HDFS、Hive、HBase、Flume、Kafka 等。
- ❑ 分布式应用系统：其实分布式应用系统很早就开始实现了，例如目前常用的邮件系统设计架构，一般都由多个数据中心组成，企业大量分支机构或较小的分散站点与数据中心的连接。分支机构需要建立自己的邮件服务器，来加快处理当地分支机构的邮件。承载相应的数据处理量，以提高邮件处理能力和收发速度。

4. 集群

大数据平台的基本特性之一，是解决海量数据的存储与计算的资源压力，提升服务器整体计算能力的解决方案。一台单独的计算机通过网络连接并构成一个群组，就具备了基本的集群特征。集群可以在付出较低的情况下，获得性能、可靠性、灵活性、扩展性、伸缩性等方面的较高收益。

服务器集群是由互相连接在一起的服务器群所组成的一个并行式或分布式系统。服务器集群中的服务器运行同一个计算任务。因此，从外部看，这群服务器表现为一台虚拟的服务器，对外提供统一的服务。尽管单台服务器的运算能力有限，但是将成百上千的服务器组成服务器集群后，整个系统就具备了强大的运算能力，可以支持大数据分析的运算负荷。根据典型的集群体系结构，企业大数据平台涉及必须使用集群架构的系统有以下几个方面：

- ❑ 数据存储集群：典型的如关系型数据库集群、NoSQL 集群、Hadoop 集群等。目前一般都会使用 MySQL 集群对关系型数据进行存储，MongoDB 则是主流的 NoSQL 类数据库集群。而 Hadoop 系列集群更是大数据平台底层存储不可或缺的一部分。
- ❑ 数据计算集群：大数据平台主流组件目前都为集群模式，例如实时计算组件 Spark、离线计算组件 HiveSQL、Map/Reduce 等。由于计算组件的部署和应用严重依赖存储组件，所以在选择时应考虑原始数据存储平台是否支持并结合计算类型。
- ❑ 高并发集群：高并发集群是大数据平台的又一大特征。由于同一时间内产生的高并发访问所产生的单台服务器资源不足的问题，典型业务场景如海量用户同一时间请求一个网站、多个系统同时请求一个数据源等，目前成熟的高并发集群组件有 Dubbo、Nginx 等。

5. 缓存

与硬件缓存所不同的是，大数据平台中的缓存主要是针对数据查询或数据交换的，当

执行高并发查询时,增加数据缓存会对查询效率有大幅提升。在此场景下,一般都会使用基于内存存储的缓存组件,例如 DB2、Memcache、Redis 等,不过也会使用基于硬盘存储的 MongoDB、Kafka 等作为缓存组件。另外,企业大数据平台中的缓存是视技术场景而选择相对应的组件,缓存主要应用的场景包括:

- ❑ 数据同步缓存:可以看看 Kafka 的设计思想,目的是通过 Hadoop 的并行加载机制来统一线上和离线的消息处理,也是为了通过集群来提供实时的消费。也就是说,在数据同步的过程中提供缓存机制,可以进行数据的第一次处理和聚合,并分发到不同数据库中。数据同步时增加缓存最大的好处是实现对数据的一次加载多次消费,减少了大量的数据加载发送所占用的数据库和硬件资源。
- ❑ 数据计算缓存:典型场景包括数据统计时所产生的临时文件的即时缓存。在进行数据挖掘或文本挖掘的过程中,由于需要做大量的数据解析、数据清洗和多次统计,会产生大量的中间数据。一个好的技术架构,是可以通过在某些部分提供缓存组件提高计算效率的(类似在 Hadoop 架构中,Map/Reduce 运行计算过程中就会产生大量的中间数据,为了解决该问题,Hadoop 的各位先驱正在不断提出新的思路)。所以在架构设计中,基于数据计算的缓存如何搭配是整个平台运行效率的一大影响因素,一般情况下会使用基于内存的 DB2 或 Memcache 进行缓存,如果缓存数据比较大也会使用 MongoDB。
- ❑ 数据查询缓存:数据查询缓存是在高并发读写的需求下产生的,而高并发读写也是“缓存”概念形成的主要驱动。主要应用场景如通过用户 ID,查找出对应的姓名、年龄、生日等信息,所以又产生了“key/value 存储结构”,例如常用的 Redis、Memcache、HBase、MongoDB 等组件的存储结构都是该格式,在大型网站中一般都可以用来缓存用户信息、订单信息和商品信息。

6. 异步

在大数据平台中的多个功能模块交互的架构设计时,最重要的是要考虑模块之间的数据传递,传递数据的过程就有两种:同步和异步。同步程序实现很简单,常见的情况是把所有功能模块编译在同一个调度流程内,请求指示调用,来执行下一个功能模块,这样的请求响应实现数据共享不难,但是同步结构的功能模块一般是不能独立运行的。异步结构的实现稍微复杂一些。因为要求异步结构的不同功能模块作为独立进程,能够独立运行,但在模块间就存在一定的进程壁垒,这就要求进程间能够实现通信以在进程间传递数据。在架构设计中包含异步的原则,使大量消耗内存的应用程序能够正常运行,并在高并发时仍然保持较好的性能。

同步和异步是相对的,一定会在平台中搭配同时存在。比如,一个同时执行多个数据查询请求时,从前端发出第一个请求后由于需要花费几分钟做即时计算,该请求就处于等待状态,等待后台返回结果。此时第二个请求已经发出,该请求只查询 1 条数据只需要花 1 秒就可以返回结果。所以在第一个请求处于等待状态时,通过异步架构可以使第二个请求也能正常运行返回结果。

在大数据平台实时查询的场景下，响应效率是最为关键的，因此大数据存储架构本身的设计需要满足最小延时的功能。在异步的处理方式下，数据首先会被获取，记录下来然后再用批处理进程进行分析。异步处理的大数据分析中遵守了捕获、存储加分析的流程，过程中数据由传感器、网页服务器、销售终端、移动设备等获取，然后再存储到相应设备上，然后再进行分析。由于这些类型的分析都是通过传统的关系型数据库管理系统（RDBMS）进行的，数据形式都需要转换或者转型成为 RDBMS 能够使用的结构类型，例如行或者列的形式，并且需要和其他的数据相连续。所以会使用到 Hive 进行存储，当数据分析开始时，数据首先从 Hive 数据仓储中会被抽出来，通过 HiveSQL 进行分析，产生需要的报告或者支撑前端的大数据应用。

7. 灾备

大数据平台灾备方案通常有两种：同城双活和本地备份，了解 Hadoop 的人都知道，其架构本身就自带本地备份方案，由于大多数企业的业务量和数据量有限，使用的该方法是最经济实惠的。而同城双活方案在容灾备份业务中是最高级别的备份方案，可实现本地与异地同时对外提供业务服务，同时实现相互备份能力。下面分别介绍这两种方案：

（1）同城双活

实现两个处在不同地域的数据中心双活模式容灾，即任何一个数据中心发生灾难时，另一个数据中心可自动接管业务。正常情况下将大数据业务分布到两个数据中心，生产和容灾中心同时对外提供服务，通过后台数据同步复制，实现两个数据中心数据的一致性。为了确保生产中心和容灾中心的数据同步不影响生产系统的性能，要求两地之间的互联网络具备较高的可靠性和足够的带宽。建议采用 FC 链路作为生产中心和容灾之间的互联网络。

两中心需同步数据包括源数据、HDFS、Hive、HBase 等，其中，源数据采用从生产中心向容灾中心同步方式，可通过事务 log 手段实现两中心源数据同步；Hive 数据采用生产中心大数据集群与容灾中心大数据集群双向同步策略，在两中心任一中心添加、修改的数据都会同步到另一个中心；HBase 数据采用从生产中心向容灾中心同步方式，通过 HBase 的 WAL（Write Ahead Log）实现同步；HDFS 数据采用从生产中心向容灾中心同步方式，借助 snapshot 对生产中心做快照，再借助 distcp 按多个 snapshot 差异信息将变化的文件同步到容灾中心。

（2）本地备份

本地大数据备份方案比较简单，主要的实现是将 HDFS、Hive、HBase 等组件的快照技术导出数据，即直接导出 HDFS 文件，包括 HBase 存放在 HDFS 的文件。值得一提的是，Hadoop 系统提供了数据压缩服务来优化磁盘的使用率，提高备份文件的传输速度。另外，Hadoop 集群中的一个文件默认存储 3 份，且分布在不同的集群节点中。

8. 自动化

大数据平台自动化越来越普遍地在企业中被采纳，因为比大数据本身更重要的是大数据

平台的分析管理能力，而这一潮流正让大数据自动化运行管理系统工具大量涌现。自动化不仅涉及大数据平台后期应用，还涉及运维、数据管理、挖掘等重要环节。自动化数据管理也应该成为其中一个重要的组成部分，它的自动化程度对于提高信息安全保障能力具有重要的意义。

- 运维自动化：其实大数据平台运维主要的核心部分就是 Hadoop 的运维，为了解决 Hadoop 运维问题，Apache 社区里已经发布了几个实用的运维工具，例如 Ambari、Mesos 等。分布式集群的管理运维，要同时解决系统的海量节点的管控问题，以及接入点的高可用问题。建议通过采用双机软件和高可用数据库，确保集群配置等信息在软件、硬件失败条件下不影响管理员对集群的有效管理。同时，需要基于灵活的架构设计，支持多种数据类型的规范化能力，以及未来可能出现的其他类型接口需求，确保大数据平台有机融入统一的管理系统。
- 自动化数据管理：在传统 IT 中对于数据的管理都是通过人工进行的，但在大数据的前提下，依靠人工的方式很难实现人工对于数据各个环节的管控，因此自动化的方式成为必要选择。自动化数据管理包括元数据管理、元数据分析、数据质量管控、数据整合管理、数据标签管理、数据资源管理、数据应用管理、数据服务管理、数据多租户管理等。
- 自动化数据挖掘：数据挖掘、机器学习、深度学习、神经网络等方法都是针对海量数据的知识提取和数据学习方法。传统的“数据智能”都是在人工选择模型、调整参数、结果校验的基础上，进行自动化的数据计算，但这离真正的“数据智能”或“人工智能”相差很远。通过对数据智能中的数据预处理、模型选择、参数调优、效果评估、部署应用等环节进行整合，同时通过建立针对模型调优的效果评估模型，以将其中关键的建模和调优部分的逻辑固化，达到自动化智能学习的不断演进。

6.4 本章小结

本章我们提供了一些数据结构化和处理模式的组合方法，来详细定义如何完整地搭建一套大数据平台的架构。接下来通过定义大数据架构来解决大数据业务问题，为解决方案中涉及的各层和组件提供一个逻辑架构。最后，还提供了来自项目实战中比较典型的示例业务问题，并对每种业务给出在架构方案中对应的组件和解决方法。

本章有以下内容需要读者重点掌握：

- 掌握大数据核心架构要素的 10 项内容，尤其是提高对前瞻性、易维护性、实用性和统一管理的关注；
- 掌握大数据架构设计模式的 8 项内容，重点是自动化实现的部分，这是未来产生数据智能的基础。

大数据技术开发

大数据技术开发相当于整个大数据项目中概要设计的部分，它上承架构，下启详细设计和开发，是整个大数据项目工作的重要枢纽和落地指导。本章将围绕大数据技术开发中的数据采集、数据存储、多维计算、功能服务、平台管理和应用域展开介绍。

7.1 数据采集

大数据技术的战略意义不在于掌握庞大的数据信息，而在于对这些含有意义的数据进行专业化处理。要对数据进行处理，首先需要将海量的各式各样的数据通过不同的数据采集渠道采集到大数据平台中进行存储。

数据采集层的主要作用是从业务数据库到大数据平台之间的数据同步，包括全量采集和增量采集两种方案。全量采集是将业务数据库中一定时间段内的所有库、表、字段和数据全部抽取并同步到大数据平台中，此方案的特点在于因为是抽取全量数据，没有字段不统一、表更新后的一致性、数据丢失等问题，但由于需要一次性抽取的数据量大，且抽取时间集中，容易造成数据库和平台集群压力过大、数据任务延误等情况。增量采集是解决快速有效采集数据，同时不会对业务数据库产生大的性能影响的另一种方案。结构化数据库的增量采集是通过解析源数据库在线日志或归档日志获得数据的增删改变化，基于解析后的日志结构化数据进行复制，再将这些变化结果写入大数据平台，实现源数据库与大数据平台的实时增量同步。

数据采集是一个复杂的过程。简单来说，数据采集可以是网页手工录入数据，也可以是扫描条形码，还可以是自动化定时数据抽取、修正或者补录等，最终为数据存储或数据分

析提供基础内容。可见无论何种情况,数据采集一定需要以下几个重要组成部分:目标、输入、处理、输出。在技术快速发展的今天,各式各样的数据采集工具和数据采集系统先后问世,数据采集进入了一个全新的时代。不同的行业、公司对于数据的存储采用了各式各样的组件,例如常见的 Oracle、MySQL、文件、缓存等,在这里我们首先对数据的类型和采集方式做一个简单的概括,如表 7-1 所示。

表 7-1 不同数据源的采集信息对比

采集场景	源数据描述	源数据存储方式
实时数据采集	实时数据接入,实时交易或者日志数据,实时性要求高	本地日志文件、互联网网页
结构化数据批量采集	大量的结构化数据批量采集	关系型数据库/文件
日志数据的采集(流式数据)	日志数据流式接入,适用于数据的流式分析场景	日志流式数据
整库、整表数据批量采集	数据仓库中整库、整表的数据	关系型数据库数据
半结构化数据采集	半结构化日志、文本文档	日志数据、互联网网页
非结构化数据采集(文档)	Word、Excel、PDF、Markdown 等	小文件形式存储
非结构化数据采集(视频、音频)	视频、音频格式文件	中大型文件形式存储

针对源数据系统中各式各样的数据类型,一个优秀的大数据云平台必须覆盖几乎所有的数据采集场景,这就需要对数据采集层进行一些复杂的组件封装,方便客户端进行数据的采集工作。

对于数据采集层来说,通常需要具有以下特征:构建应用系统和分析系统的桥梁,并将它们之间的关联解耦;支持近实时的在线分析系统和类似于 Hadoop 之类的离线分析系统;具有高可扩展性,即当数据量增加时,可以通过增加节点进行水平扩展。实时数据采集相比传统的数据采集技术,在实时性方面做到秒级甚至毫秒级,对于日志等信息的处理更加迅速与快捷。如何做到实时抽取准确分析,大幅提升数据同步效率,并扩展数据采集的功能,是数据采集层的技术关键之所在。

7.1.1 批量采集

批量采集即批量离线数据采集,数据源包含结构化数据处理能力(可以是文本抽取、JDBC 抽取、Oracle 抽取、DB2 抽取、Hive 抽取等),半结构化数据处理能力(XML 抽取),非结构化数据处理能力(HBase 抽取等),可利用 Sqoop 组件实现数据的批量抽取、转换、加载。

基于 Sqoop 的数据整合工具 Loader 是在开源 Sqoop 组件 1.99.x 版本的基础上进行了一些扩展,实现大数据平台与关系型数据库、文件系统之间交换“数据”“文件”,既可以将数据从关系型数据库或者文件服务器导入大数据平台的 HDFS/HBase 中,同时也支持反过来从 HDFS/HBase 导出到关系型数据库或者文件服务器中。Loader 工作方法如下。

(1) 通过 MapReduce 实现并行执行和容错

Loader 通过 MapReduce 作业实现并行的导入或者导出作业任务,不同类型的导入导出

作业可能只包含 Map 阶段或者同时包含 Map 和 Reduce 阶段。同时利用 MapReduce 实现容错，在作业任务执行失败时，可以重新调度。

(2) 数据导入 HBase

在 MapReduce 作业的 Map 阶段中从外部数据源抽取数据。在 Reduce 阶段中，按 Region 的个数启动同样个数的 Reduce Task。Reduce Task 从 Map 接收数据，然后按 Region 生成 HFile，存放在 HDFS 临时目录中。在 MapReduce 作业的提交阶段，将 HFile 从临时目录迁移到 HBase 目录中。

(3) 数据导入到 HDFS

在 MapReduce 作业的 Map 阶段中从外部数据源抽取数据，并将数据输出到 HDFS 临时目录下（以“输出目录-ldtmp”命名）。在 MapReduce 作业的提交阶段，将文件从临时目录迁移到输出目录中。

(4) 数据导出到关系型数据库

在 MapReduce 作业的 Map 阶段，从 HDFS 或者 HBase 中抽取数据，然后将数据通过 JDBC 接口插入临时表（Staging Table）中。在 MapReduce 作业的提交阶段，将数据从临时表迁移到正式表中。

(5) 数据导出到文件系统

在 MapReduce 作业的 Map 阶段，从 HDFS 或者 HBase 中抽取数据，然后将数据写入到文件服务器临时目录中。在 MapReduce 作业的提交阶段，将文件从临时目录迁移到正式目录中。

(6) Loader 主要增强特性

- ❑ 具备多种接入协议适配能力：数据源接口协议支持 SFTP、FTP、关系型数据库 JDBC/ODBC 多种接口。
- ❑ 具备分布式接入数据能力：导入数据到 HBase 和 Phoenix，支持 Bulkload 分布式导入模式。
- ❑ 具备多种导出协议适配能力：数据导出目的端接口协议支持 HBase、HDFS、Phoenix、SFTP、FTP。
- ❑ 支持所有类型文件导入：开源导入工具 Sqoop 只支持文本文件和 sequence 格式文件，增强 Loader 工具支持所有文件类型。
- ❑ 支持对多文件合并：如输入文件为海量文件，可以合并为 n 个文件（ n 值可配）。
- ❑ 支持导入（导出）时对文件进行编码转换：支持对文件进行转换编码格式，支持的编码格式为 JDK 支持的所有格式。

7.1.2 增量采集

增量采集的需求场景非常多，典型的是关系型数据库中的数据较为庞大，并且数据的写入和更新操作非常频繁，例如企业的订单数据库、财务数据库等。由于订单的状态实时变化会导致数据库中数据的更新和删除的频繁操作，利用传统的批量抽取操作将会导致数据状态

的严重滞后,从而影响分析结果。由此需要增量数据采集方案将数据实时同步到 Hadoop 平台中,在此我们采用笔者自主研发的另一款组件 DataIn。该组件通过对 Goldengate、Canal 和 DMETL/DMHS 组件的封装,DataIn 产品将实现 Oracle、MySQL 与达梦数据库中的数据实时同步到 Hadoop 平台的功能。

该组件工作流程是实时获取 Oracle、MySQL 和达梦三大主流结构化数据库中的数据变化,同时将实时数据推送到分布式消息中间件 Kafka,同时将 Kafka 中的数据实时写更新到对应 Hive、HBase 表中,实现数据源端到大数据平台之间的数据实时同步机制(数据的实时同步时效性可以毫秒级)。此组件的特点在于实时同步当时的增量数据,使 Hadoop 集群在数据同步部分的压力均匀分摊,同时可以实时应用。具体工作流程如图 7-1 所示。

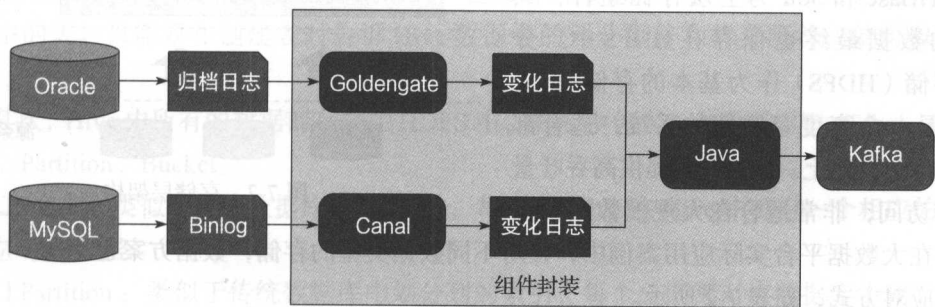


图 7-1 增量采集工作流程

全面并且高效的数据异常处理机制,以便在数据稳定性与一致性出现问题时,能够快速处理问题数据并恢复数据同步链路。通过获取数据库的归档日志,并将其转化成数据的变化日志,实时读取到数据缓存中,再由数据缓存分发到各个存储组件中。整个流程需要保证很高的稳定性与数据收集的顺序性,支持至少每分钟 10 万次量级的数据变化,保证了大数据量吞吐的需求。

7.2 数据存储

大数据平台用于处理低价值海量结构化数据、半结构化与结构化数据;其与数据仓库协同,支撑数据应用系统,弥补数据仓库的不足。从控制架构复杂度的角度考虑,应用系统应选择其主要数据源作为数据整合者,不同时与两者直接交互数据。

传统数据仓库架构制约了数据存储能力和计算能力,为了应对这些问题,基于 Hadoop 的分布式数据仓库已经成为数据存储中广泛采用的事实标准。但 Hadoop 在 SQL 兼容性和复杂逻辑即时查询的情况下还不能完全替代传统数据仓库,所以一般在传统数据仓库的场景下,使用前期共存后期逐步替代关系,即传统应用继续基于传统数据仓库技术,大数据相关的新应用采用大数据平台(大数据应用指需要做混合数据处理和传统数据仓库在规模、成本、

效率方面都无法满足的应用)。数据仓库承载及时性、准确性要求高的核心事务型关键应用，大数据应用开发平台承载数据仓库痛点业务迁移和大数据创新业务，这两者都会通过这种混搭架构实现。

在企业业务方面，如 7.1 节所述，企业中存在各种各样的数据类型，例如结构化、半结构化和非结构化数据，针对不同的数据类型，大数据平台有针对性地采用了不同的存储组件，存储层架构示例如图 7-2 所示。

架构中对于不同的数据类型规划了四种存储进行分别存储。其中最底层的存储组件为 HDFS (Hadoop 分布式文件系统)，Hive、HBase 和 Solr 为上层存储组件，其存储的数据最终也保存在 HDFS 中。分布式存储 (HDFS) 作为基本的存储组件，HDFS 是一个高度容错性的系统，适合部署在廉价的机器上。HDFS 能提供高吞吐量的数据访问，非常适合在大规模数据集上应用。在大数据平台实际应用案例中，针对不同数据类型的存储，数据方案也不同，应采取不同的应对方式，如表 7-2 所示。

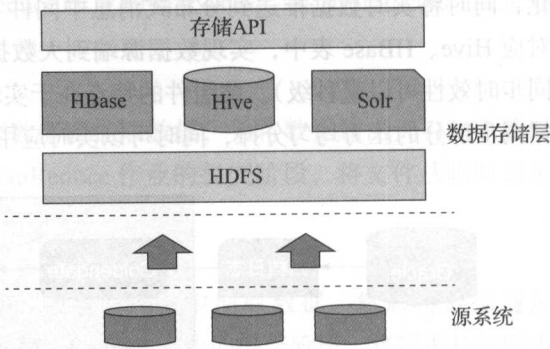


图 7-2 存储层架构

表 7-2 不同数据的适用场景对比

数据类型	数据集名称	适用场景
表数据	离线表数据集	存储结构化表数据，用于离线计算，响应时间为分钟级
	在线表数据集	存储结构化表数据，用于近实时计算，响应时间为秒级
	实时表数据集	存储结构化表数据，用于实时计算和查询，响应时间为毫秒级
文件数据	文本文件数据集	存储半结构化 / 非结构化数据文本数据
	文档文件数据集	存储非结构化文档数据，例如 Word、Excel、PDF 等文档型数据
	媒体文件数据集	存储非结构化媒体数据，例如图片、视频、音频等媒体数据

7.2.1 HDFS 文件存储引擎

运行在 HDFS 上的应用具有很大的数据集。HDFS 上的一个典型文件大小一般都在 G 字节至 T 字节级别。因此，HDFS 被调节以支持大文件存储。它应该能提供整体上高的数据传输带宽，能在一个集群里扩展到数百个节点。一个单一的 HDFS 实例能支撑数以千万计的文件。换句话说，其实 HDFS 本身就是为了解决海量数据文件的低成本高效存储而产生的，不管是文本、文档、图片还是视频文件的存储，使用 HDFS 是最佳解决方案。

另外，HDFS 的可配置性极高，同时，它的默认配置能够满足很多的安装环境。多数情况下，这些参数只在非常大规模的集群环境下才需要调整。唯一遗憾的是，HDFS 在存储的数据访问权限方面处理得过于简单了，仅仅启动 NameNode 的用户才被视为 HDFS 的超级用

户。不过 HDFS 后续的版本将会支持网络验证协议（例如 Kerberos）来对用户身份进行验证和对数据进行加密传输。

7.2.2 Hive 数据存储引擎

Hive 数据存储引擎是大数据平台重要的结构化数据的存储与计算组件，采取批量抽取方式进行数据的加载。由于 Hive 的类 SQL 功能和类数据库功能，它向非编程人员开放了大数据 Hadoop 生态系统，故常被描述为一个构建于 Hadoop 之上的数据仓库基础架构，这是一种部分真实的表述，但在创建事实表和维度表时，它更关乎设计而不是技术。尽管如此，使用 Hive 构建离线数据仓库，用来存储单表海量数据依然是最优的方案。

首先，Hive 没有专门的数据存储格式，也没有为数据建立索引，可以非常自由地组织 Hive 中的表。只需要在创建表时告诉 Hive 数据中的列分隔符和行分隔符，就可以解析数据了。

其次，Hive 中所有的数据都存储在 HDFS 中，Hive 中包含 4 种数据模型：Table、External Table、Partition、Bucket。

- Table：类似于传统数据库中的 Table，每一个 Table 在 Hive 中都有一个相应的目录来存储数据。
- Partition：类似于传统数据库中划分列的索引，每个分区都对应数据库中相应分区列的一个索引。表中的一个 Partition 对应表下的一个目录，所有的 Partition 数据都存储在对应的目录中。
- Bucket：在指定列进行 hash 计算时，会根据 hash 值切分数据，使每个 Bucket 对应一个文件，目的是便于并行处理。
- ExternalTable：指向已存在 HDFS 中的数据，可创建 Partition。和 Table 在元数据组织结构相同，但在实际存储上有较大差异。Table 创建和数据加载过程，可以用统一语句实现，实际数据被转移到数据仓库目录中，之后对数据的访问将会直接在数据仓库的目录中完成。数据存储在 Location 后面指定的 HDFS 路径中，并不会移动到数据仓库中。

7.2.3 HBase 列式存储引擎

相对于传统的行式存储格式，列式存储引擎因具有更高的压缩比和更少的 IO 操作而备受青睐。HBase 适合大量插入同时又有读的情况。输入一个 key 获取一个 value 或输入一些 key 获得一些 value。所以适合在线表数据集的存储，以便于解决在线数据的实时读取问题。

HBase 的瓶颈是硬盘传输速度，它的存储方式像是日志文件一样，是批量地往硬盘中写，通常都是以文件形式读写。这个读写速度，就取决于硬盘与机器之间的传输有多快。需要注意的是，HBase 的所有操作都是追加插入操作，它可以往数据中 insert，也可以 update 一些数据，但 update 实际上也是 insert，只是插入一个新的时间戳的一行。delete 数据，也是 insert，只是 insert 带有 delete 标记的一行。

❖ HBase 是以列作为元素存储的。同一个列的元素会挤在一个块。当要读某些列时，只需要把相关的列块读到内存中，这样读的 IO 量就会少很多。通常，同一个列的数据元素格式都是相近的，这就意味着，当数据格式相近时，数据就可以做大幅度的压缩。所以，列式数据库在数据压缩方面有很大的优势，压缩不仅节省了存储空间，同时也节省了 IO。这一点，可利用在当数据达到百万、千万级别以后，通过数据查询之间的优化来提高性能。

7.2.4 MySQL 关系型数据存储引擎

❖ 大数据平台中存储层仅仅有了 Hadoop 系列组件就可以抛弃关系型数据库了吗？答案当然是否定的。数据库作为一个非常基础的系统，任何一家企业都会使用，以上三种存储组件主要是解决各种类型海量数据的存储，如果某些业务线只是 MB 级别的数据量，使用以上三者显示是杀鸡用牛刀了，所以 MySQL 是小数据量存储和复杂逻辑关系数据存储的最优解决方案。最典型的应用场景如员工名单、财务数据的存储，二者数据量小，有一定的逻辑关系，而且需要进行即时查询，使用 MySQL 就可以快速地解决问题了。

但是随着企业关系型数据的不断积累，单机版 MySQL 往往无法满足需求，自然而然我们就需要考虑结合 Hadoop 架构进行数据存储架构的二次优化，或者直接引入分布式的 MySQL 解决方案去支撑未来关系型数据存储的发展。

7.3 多维计算

❖ 计算层是从大量的原始数据中抽取有价值的信息，即数据转换成信息的过程。主要对所输入的各种形式的数据进行加工整理，这一过程包含对数据的收集、存储、加工、分类、归并、计算、排序、转换、检索和传播的演变与推导。计算层包括两大基本因素：数据处理和数据管理。数据处理是指对数据进行收集整理、组织、存储、维护、检索、传送等操作，该部分也是后续进行数据管理时的必要部分。数据处理涉及的计算比较简单，需要根据业务的需求来编写应用程序加以解决，加工计算会根据具体的业务来定制。而数据管理则比较复杂，是针对数据的爆炸性增长和多种繁杂类型进行统一处理，表 7-3 所示是大数据平台计算层需要考虑的多种场景和对应的实现框架参考（实现框架在第 5 章中已详细介绍）。

表 7-3 不同业务计算场景说明

业务场景	场景说明	实现框架参考
离线批处理计算	全量数据的离线批处理计算，计算数据流巨大，响应时间在分钟级	MapReduce、Hive
近实时交互查询	少量数据的交互式查询场景，响应时间在秒级	Impala
实时处理计算	少量数据的实时查询处理，响应时间在毫秒级	Spark、HBase
流式处理计算	少量数据的逐条或者时间窗口批处理，响应时间在毫秒级	Storm、Spark Streaming
全文检索查询	全量数据的全文检索查询，响应时间在毫秒级	Solr、ElasticSearch、Lucene

❖ 数据处理与数据管理是相联系的，数据管理技术的优劣将对数据处理的效率产生直接

影响。而数据库技术就是针对该需求目标进行研究并发展和完善起来的计算机应用的一个分支。从数据管理角度而言，不仅要使用数据，而且要有效地管理数据。因此，一般需要一个通用的、使用方便且高效的管理软件，把数据有效地管理起来。下面会介绍一个笔者自主研发的主要针对计算层的大数据应用服务工具 Paper。该工具对数据处理进行了规范化的设计，使数据管理成为固定的处理流程，从数据源、数据处理、挖掘管理、展示输出等环节为用户提供简单方便的大数据计算服务，使用者只需以界面拖拽的方式进行数据处理流程的组合即可完成数据处理流程的配置。同时兼容不同版本的 Hadoop 平台，且支持以 RESTful 接口的方式配置数据处理流程。数据流程配置样例如图 7-3 所示。

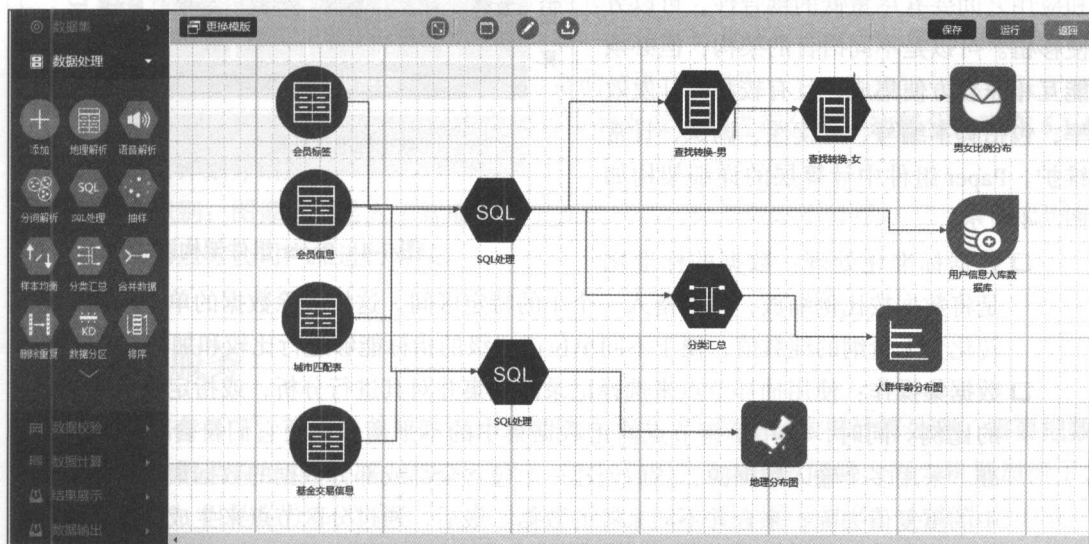


图 7-3 数据流配置示意图

如图 7-3 所示：Paper 固化了数据处理的流程，所有的数据处理遵循“数据接入”→“数据处理/校验/转换”→“图形展示/数据输出”。具体应用流程如下：

- 数据集：将要处理的数据对象定义为数据集，可以是一个语音文件、一个本地 log、数据库中的某个表或数据流中的某个节点产生的结果。
- 数据校验：对数据集的分布、异常和缺失进行校验和分析。
- 数据处理：对数据集进行抽样、转换、合并、删除、解析等数据预处理工作，主要是对数据格式、缺失值、异常值、记录、字段等进行处理，以便得到符合后期数据计算和处理所需的数据。
- 数据计算：通过大数据挖掘和机器学习模型对海量数据进行挖掘和学习，得到潜在的数据知识和规律。
- 数据展示：通过多种在线图表等形式展示结果，以得到有效的结论理解和输出展示。
- 数据输出：将数据流的运算结果输出，以便于集成到数据仓库、应用系统中做进一步处理。

虽然 Paper 组件可以使我们快速上手操作计算层，但是在建设计算层时我们应该注意的是，通过计算层来统一计算任意结构的数据，不仅包括数据库，也包括非数据库的 Excel/Txt/XML。其中对最常见的结构化数据的计算是重点。统一地进行不同种类数据源之间的相互计算，不仅包括异种数据库之间的计算，也包括数据库和非数据库之间的计算。数据库和计算层、计算层和应用之间要有尽量低的耦合性，可以方便移植。可以是不同语言的架构，但必须能互相兼容方便集成，且有较高的开发效率，包括脚本编写、可读性、调试、日常维护。Paper 组件中计算层的详细架构如图 7-4 所示。

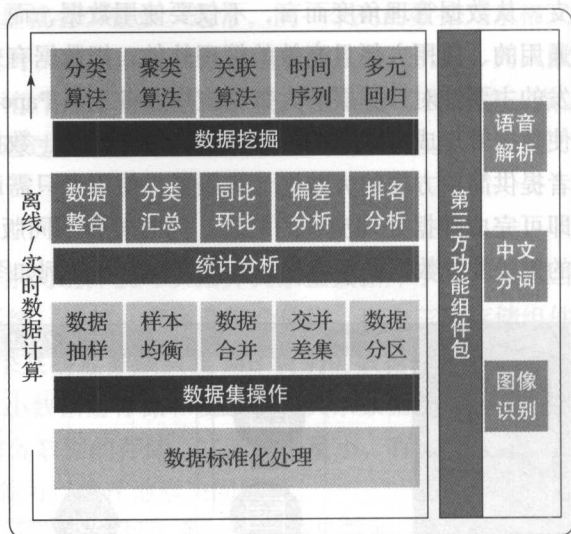


图 7-4 Paper 组件架构图

□ 数据标准化处理：数据的标准化

是将数据按比例缩放，使之落入一个小的特定区间。这样去除数据的单位限制，将其转化为无量纲的纯数值，便于不同单位或量级的指标能够进行比较和加权。

□ 数据集操作：使用抽样节点来选择记录的子集并对其进行分析，或指定要丢弃的记录的比例。同时，使用平衡节点修正数据集中的不平衡，以便它们符合指定的检验标准。采用多个输入数据源，然后创建一个包含全部或部分数据的数据集，将指定变量中的重复值去除，来合并不同来源的数据。最后，通过分区节点来生成分区字段，将数据分割为单独的子集或样本，以供模型构建的训练、测试和验证阶段使用。

□ 统计分析：在统计分析操作时，数据整合汇总是一个必要的步骤，用来减小数据集从而减少无用的计算。排序节点可根据一个或多个字段的值，按照升序或者降序对记录进行排序。离散化指把连续型数据切分为若干“段”（也称 bin），是数据分析中常用的手段。切分的原则有等距、等频、优化，或根据数据特点而定。同时，时间序列水平中无法解释的变量称为离群值。这些观测值与序列中的其他值不一致，可能会显著影响分析，从而影响时间序列模型的预测能力。以上几种方法为统计分析时常用到一些手段。

□ 数据挖掘：分类机器学习算法用于结果分类预测，根据预测结果取值数分为二元或多元，其结果具有非常强的可解释性、理解性和应用性，例如响应和不响应，高、中、低等，此类算法广泛应用于计划、排期、预算、目标制定等战略执行活动之前。聚类算法是一种监督式算法，通常没有结果值可供参照，且预测的结果通常是具有无特征意义的分类类别，适用于做群体特征分析，其结果为非人为定义的群体标签值。关联算法是反映的是提取一个特定事件，计算该事件和其他事件之间依赖或关联的知识，

应用于类似人与人之间、事物与事物之间、人与事物之间的某种关联关系。时间序列算法属于预测的一种，是将同一统计指标的数值按其发生的时间先后顺序排列而成的数列，主要目的是根据已有的历史数据对未来进行预测，具有很强的适应性和移植性，能对时间规则中的周期性波动、季节性波动进行很好地拟合。回归算法是确定两种或两种以上变数间相互依赖的定量关系的一种统计分析方法，来判定某种情况发生的可能性。比如，某用户购买某商品的可能性，某患者患有某种疾病的可能性，以及某广告被用户点击的可能性等。值得注意的是，其结果并非数学定义中的概率值，不可以直接当作概率值来用。

- ❑ 语音解析：语音解析功能可将语音文件转化成文字，为后续数据处理提供半结构化文字数据。配合语音解析和分词解析，可将一段语音转化成不同文字标签的词频统计结果，并通过角色、字段、时间等提供丰富的“语音标签”。
- ❑ 中文分词：词法分析系统即对指定列对应的文章内容进行分词，这部分内容在 5.1.4 节中已详细介绍过。
- ❑ 图像识别：图像识别是人工智能的一个重要分支，在大数据平台建设中一般会使用专业的第三方公司产品进行对接。

7.4 功能服务

在大数据背景下，不可避免地面临着大规模的挑战。大规模的数据计算处理，需要把数据分布到多台机器并行处理。在单机环境下，往往不需要考虑失败问题，因为机器崩溃了，程序无法恢复。但是在分布式环境下，机器数量很大，多台机器需要协作，局部失败的概率变得很高，例如：硬件上某台机器“挂了”，其上运行的任务都“挂了”；网络上交换机或路由器崩溃；计算节点磁盘空间不足或内存溢出；数据在传输中出错或网络中断；等等。在分布式环境下，这些问题变成“家常便饭”，系统应该有能力从这种局部失败中恢复，用户可以不关心这些错误，继续正常工作，提供这种“弹性”是软件工程面临的巨大挑战。

为了应对上述问题，大数据平台从设计之初就是为公有云服务为目标，其中多租户、数据安全、数据调度、数据路由、数据挖掘算法均以服务的方式为客户提供服务；同时以服务的方式支持高度兼容标准语法的 SQL 处理以及 UDF 的使用，便于数据分析人员将其代码从传统数据库向分布式数据平台平滑迁移。

和其他的云服务产品类似，大数据平台也采用 HTTP 的 RESTful Service，客户端可以直接调用平台提供的服务来完成开发任务，大大降低了客户端的开发复杂度，并且增强了客户端的稳定性、扩展性和易用性。服务层以接口的服务化为基础，服务化作为目前最为流行的分布式架构开发方式，具有以下几点好处：

(1) 零侵入

在我们实际设计服务的过程中，很希望这个服务框架或者服务化对上层应用的侵入尽可

能小，但百分百零侵入是做不到的。利用配置化的方式，即使用 Spring 框架，通过配置和扩展、消费、发布实现零侵入，其实配置本身也是一种侵入，是代码的一部分。但是它有个好处，就是我们改一条代码总比去编译它强。也就是说，我们尽量少地开放服务框架的 API 给上层，而开放的都是一些特性和配置化的方式，或者用注解也可以。

（2）容错和路由

先来讲路由。路由其实就是客户端调用接口的一个路由策略，常见的路由策略包括随机、轮询、最小连接优先、最小负载优先等，类似一个负载均衡器，可以在高并发的环境下对服务调用进行均衡，避免对底层核心的资源造成过大的压力。

容错是保证服务高可用的关键，当一个服务调用被路由分配到一个服务提供者后，如果该服务提供者暂时不可用，就需要容错机制来对服务进行容错。我们可以通过服务的无状态加上集群容错，去真正地把底层这种异常整个给屏蔽掉，只要集群有至少一个可用的服务，该服务调用就能成功，而且服务调用方也不关心到底哪个服务提供者能用，哪个不能用。

（3）故障隔离

通过服务化以微服务的方式提供系统中的服务，服务管理统一采用服务注册、发布、订阅和调用流程，使服务和服务之间互相独立，进而在发生故障时不会相互干扰导致服务不可用。

（4）服务治理

通常来说，为了和执行工作组一起工作而一致同意建立的工作指南就是治理。具体来说，治理包括以下方面：

- 建立授权的责任链。
- 度量评估的有效性。
- 指导组织建立满足其目标的策略。
- 控制机制以确保遵从性。
- 进行沟通以使所有相关方都获得通知。

在服务层中，服务使用者和服务提供者运行于不同的进程中，由不同的部门开发和管理，为了两方能在一起工作，需要进行大量的协调工作。为了服务层能成功，多个应用程序需要能共享相同的服务，这意味着它们需要进行协调，以便共享和重用这些服务。这些就是治理问题，比采用独立应用程序（甚至包括使用可重用代码和组件时）要复杂得多。基于以上设计思路，在大数据平台中的重要性占比越来越大，如图 7-5 所示为在实际项目中服务层的应用架构示例。

此示例中，大数据平台服务层采用分层架构来实现，分别为数据网关层、数据服务总线层和功能组件层。三个层次分工明确，数据网关层负责 REST 和 JDBC 标准接口的封装和实现；数据服务总线层负责服务化接口的注册、治理、调用编排以及服务文档的管理；功能组件层负责对底层大数据平台中存储、计算、导入、导出等核心功能进行抽象和封装。

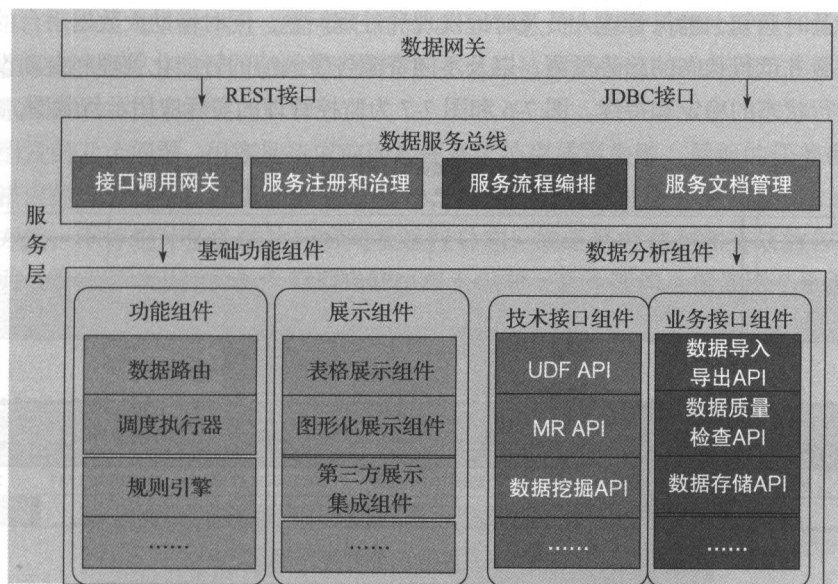


图 7-5 服务层的应用架构

7.5 平台管理

为了解决上述问题，平台管理层会通过服务器、Hadoop 集群、计算任务等大数据平台各个技术进行统一管控，共分为三大层面的管理：监控管理、调度管理、权限管理。

7.5.1 监控管理

大数据平台内部的软件资源复杂，需要支持统一化的一体监控模式，监控管理的主要功能是针对平台中各子系统以及各功能模块提供统一的运行监控服务，包括服务的提供状态监控、服务的使用状态监控、系统的运行状态监控等。建立集成化的资源运行状态监控管理系统，实现平台运行状态和信息化资源的统一化、可视化、可控化管理。

同时，需要支持对 Hadoop 架构体系内所有软件模块和系统（HBase、HDFS 等）、关系型数据库（MySQL 集群等）、接入共享接口、中间件系统、前端应用等软件资源的监控和管理，支持 SNMP、SSH/Telnet、JMX、JDBC 等监控方式和协议；支持大数据平台内部的各种传统关系型数据库和数据库集群、Hadoop 架构中 HDFS/HBase 等的状态、数据库状态、各类中间件资源状态、平台内部和前端应用状态、各类内外部的接口状态的实时监控；支持对数据接入和共享接口的运行监控；支持各类监控资源告警阈值的人工和自动调整；支持告警信息的短信、邮件通知以及声光报警通知。

另外，对平台所涉及的所有服务和数据资源进行实时不间断的 7×24 小时的监控和管理，合理设定资源状态的预警阈值，确保任何资源状态异常的及时通报和展示，通过声光报警、短

信、邮件的及时通知，确保管理人员及时地发现和解决问题。同时借助大数据平台的优势，实现平台内部服务运行状态的趋势预测，以及不同资源告警阈值的智能化管理和自动设定，做到平台内部运行状态的稳定和持续。图 7-6 和图 7-7 为监控管理的实际应用示例截图。

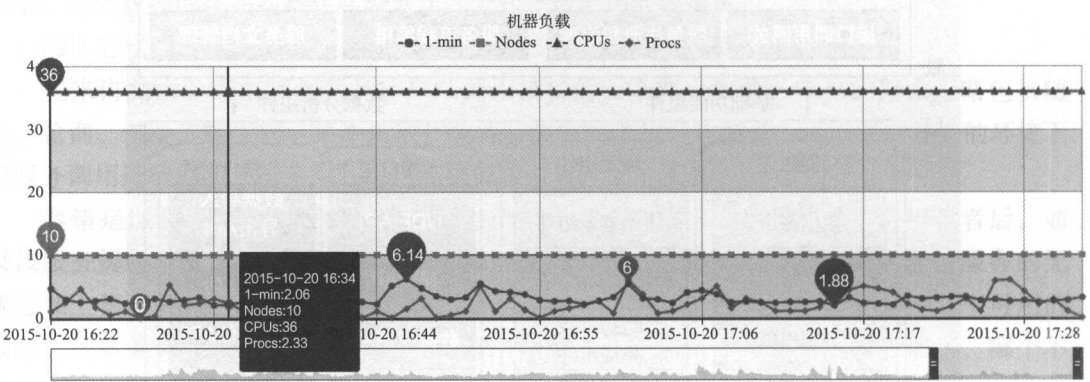


图 7-6 机器负载监控图



图 7-7 整体集群监控概览

7.5.2 调度管理

大数据平台技术框架支持的开发语言多种多样，开发人员的背景差异也很大，这就使得很多不同类型的程序（任务）运行在大数据平台之上，例如 MapReduce、Hive、Pig、Spark、

Java、Shell、Python 等。这些任务需要不同的运行环境，并且除了定时运行，各种类型之间的任务存在依赖关系，目前各业务的数据任务基本都是靠 Crontab 定时调度，各个任务之间的依赖仅靠简单的串行来实现。

由此引发的几个问题：很容易造成前面的任务未结束或者失败，后面的任务也运行起来，最终出现错误的分析结果；任务不能并发执行，增加任务执行的整体时间窗口；任务管理和维护很不方便，不好统计任务的执行时间及运行日志；缺乏及时监测和有效的告警；我们通过大数据平台管理层中设计的调度管理来解决以上问题，具体架构示例如图 7-8 所示。

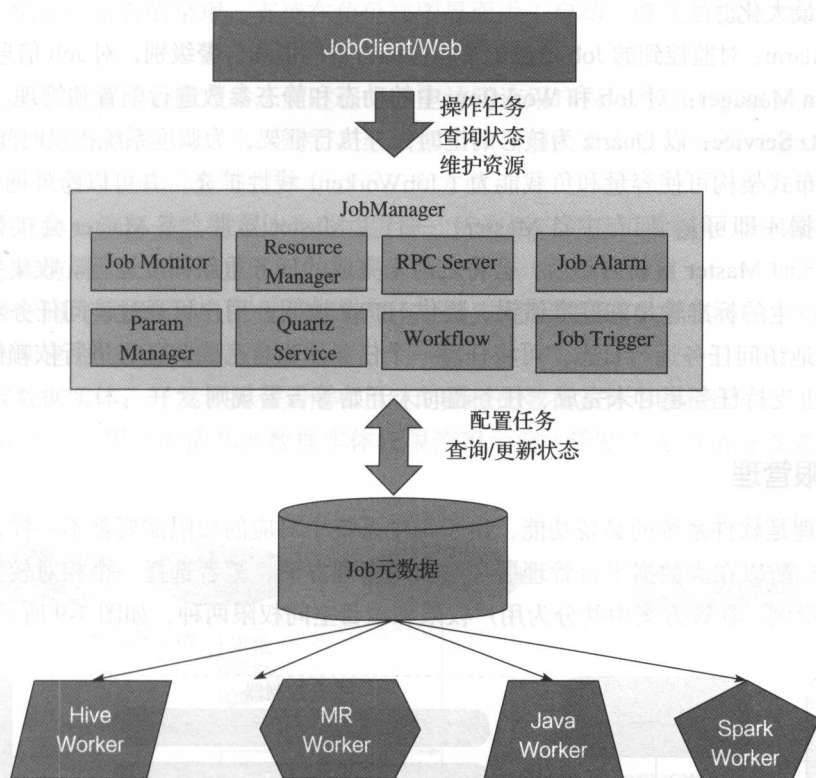


图 7-8 调度管理架构

详细模块介绍如下：

- JobManager：为整个调度系统的 Master，提供 RPC 服务，接收并处理 JobClient/Web 提交的所有操作；与元数据通信，维护 Job 元数据；负责任务的统一配置维护、触发、调度、监控。
- Job Monitor：监控正在运行的 Job 状态、监控任务池、监控等待运行的 Job。
- Worker：调度系统的 Slave，从任务池中获取 Job，负责启动并收集 Job 的执行状态，维护至元数据库。
- JobClient/Web：调度系统客户端类，前端界面提供给用户，用于任务的配置、管理、

监控等。

❑ 任务元数据：目前使用 MySQL，保存 Job 的配置、依赖关系、运行历史、资源配置、告警配置等。

❑ Workflow：提供 Job 之间依赖关系的配置，系统支持以 Workflow 的形式进行 Job 的依赖运行。

❑ Job Trigger：提供 Job 或者 Workflow 的触发器配置，目前支持时间和条件触发两种形式。

❑ Resource Manager：负责探测底层 Hadoop YARN 中剩余资源，使 YARN 中的资源利用率最大化。

❑ Job Alarm：对监控到的 Job 信息配置相应的告警阈值和告警级别，对 Job 信息进行告警。

❑ Param Manager：对 Job 和 Workflow 中的动态和静态参数进行配置和管理。

❑ Quartz Service：以 Quartz 为核心的定时任务执行框架，为调度系统提供时间调度功能。

通过分布式架构可使容量和负载能力（JobWorker）线性扩充，并可以跨外网部署（只需能访问元数据库即可）。拥有主备 Master，一旦主 Master 异常，备 Master 会接替主 Master 提供服务。同时 Master 重新启动后，会将之前未完成的任务重新调度运行。收集并记录任务运行过程中产生的标准输出和标准错误，提供 HTTP 访问，用户可通过访问任务对应的日志 URL 来方便地访问任务运行日志，可将任意一个任务作为自己的父任务进行依赖触发。除了失败告警，也支持任务超时未完成、任务超时未开始等告警规则。

7.5.3 权限管理

权限管理是软件系统的必备功能，由于每个系统中对应的权限需要都不一样，没有一个绝对的标准。所以在大数据平台管理层中的权限管理方面，笔者选择一个相对较为典型的权限方案作为示例，在该方案中共分为用户权限和项目空间权限两种，如图 7-9 所示。

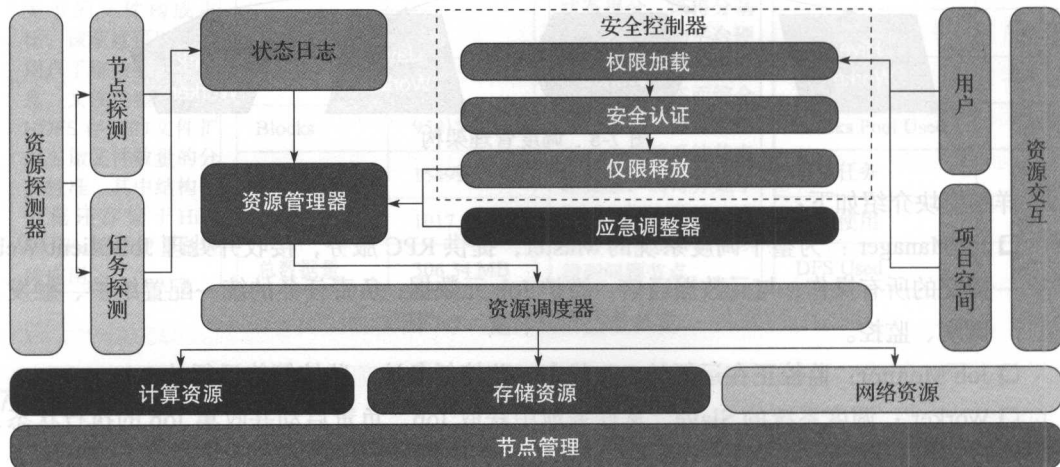


图 7-9 权限管理架构

1. 用户权限

用户按角色职能的不同,分为系统管理员、项目空间管理员、开发人员和运维人员四类角色。系统管理员负责整个系统的管理,管理控制台界面为系统管理员提供完整的有效系统支撑。比如,新增、修改、锁定及注销用户;对多项目空间环境进行创建、修改、撤销。项目空间管理员负责每个项目下的开发团队,根据需要指定其在平台上的执行账号。在开放平台上实现账号权限管理、资源控制。开发人员在项目空间内,主要负责具体应用的开发,在平台上提供统一可视化开发工具做程序开发。运维人员则负责日常程序的监控与运行维护,提供系统运行状态的呈现,支持在角色视图界面手工启动、终止数据处理、数据分发等程序。

同时,用户在进行数据产品开发过程中只能访问该用户权限范围下的数据实体,如果需要访问当前没有赋权的数据实体,则需要给数据实体当前所有者进行权限申请,审批通过后即可访问。开发者可访问的数据实体有以下三个方面:

- 公共实体:该部分数据实体为公共资源,所有用户都有可读权限,开发者无需申请,可直接访问。
- 项目空间内数据实体:开发者拥有对应项目空间下的所有数据实体的可读权限,用户自己创建的实体还有写权限。
- 赋权数据实体:开发者拥有其他项目空间赋权的数据实体可读权限。

图 7-10 所示为用户申请其他数据实体权限流程示例(开发者 A 申请开发者 B 的数据实体权限)。

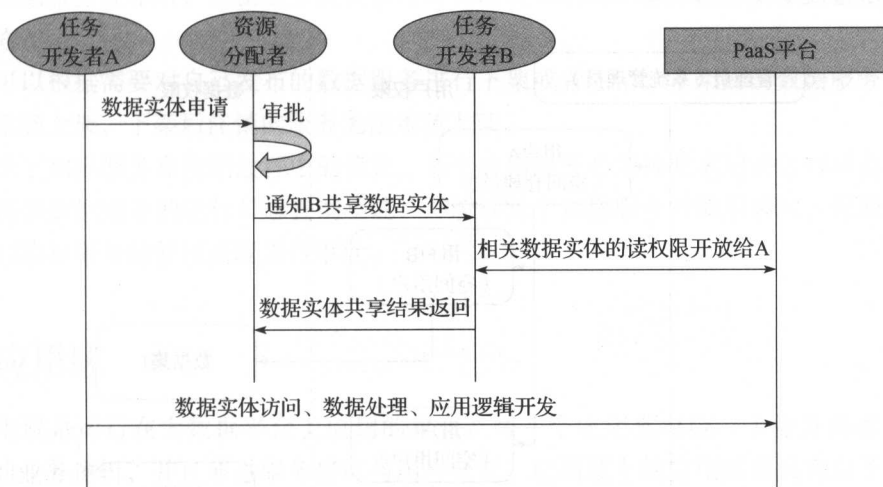


图 7-10 权限流程

2. 项目空间权限

项目空间为每个用户分配一个账号。用户账号在使用平台时,必须开通项目空间、分配

逻辑独立的资源。所分配资源包括但不限于 Hadoop 平台的 Hadoop 存储容量、HDFS 属主目录、计算资源队列，以及 FTP/DB 数据域、模型层次 / 数据域读写权限等。这些资源在逻辑上为指定项目空间专有，只能由该项目空间访问（数据实体资源的授权访问除外）。

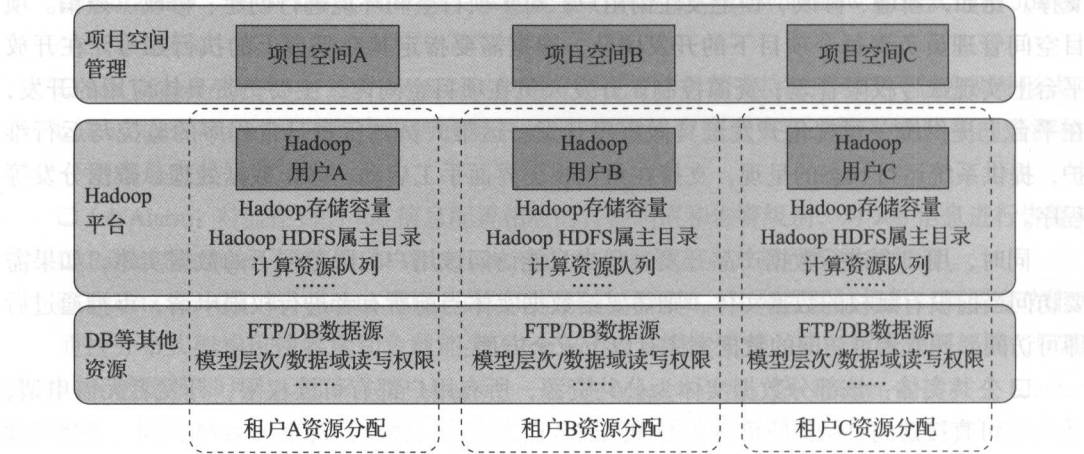


图 7-11 项目空间权限架构

项目空间资源管控是充分利用大数据平台的分析能力，帮助开发人员合理申请和优化资源，同时帮助系统管理员管控各项目空间资源，使得资源池的资源能为更多开发人员服务。在系统中用户分为 3 种角色，分别为系统管理员、空间管理员和普通开发者，他们的关系如图 7-12 所示。

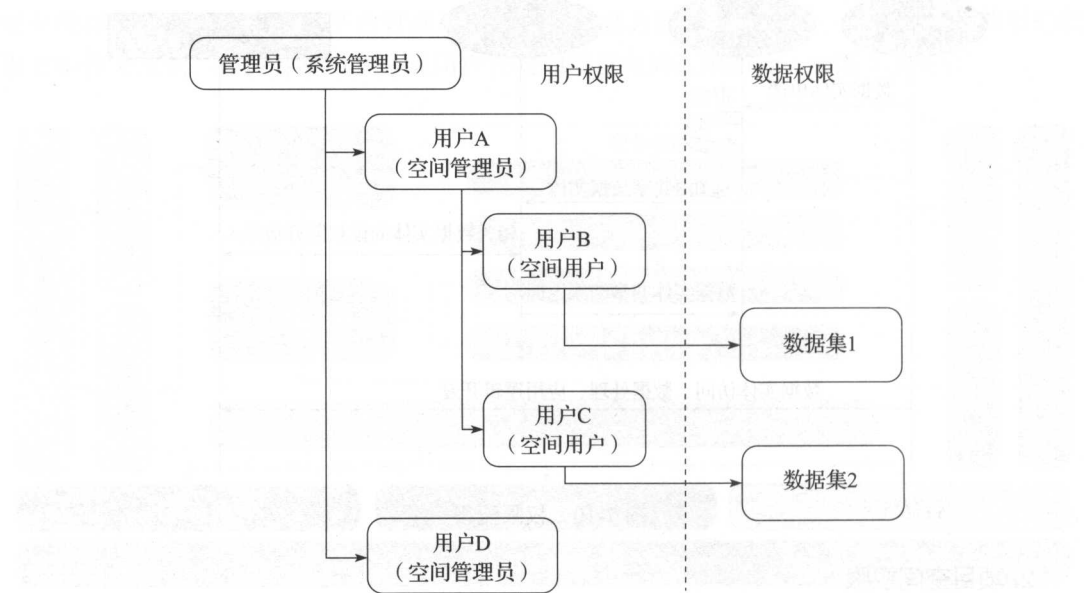


图 7-12 权限关系

首先管理员创建用户 A 和项目空间 A，并且将用户 A 设置为项目空间 A 的空间管理员，接下来系统管理员创建用户 B，并将用户 B 的角色设置为普通空间用户。随后项目空间 A 管理员用户 A 就可以将用户 B 分配到项目空间 A 中，那么用户 B 默认就有了在项目空间 A 中操作数据集的权限了。但是此时用户 C 虽然跟用户 B 处在同一个项目空间中，但是无法访问用户 B 创建的数据集，要想访问只能由用户 B 共享给用户 C 或者共享给空间 A。数据权限分为数据集的创建、删除、修改以及共享，用户可以将数据集共享给用户或者其他项目空间。

同时，为了维护服务的发布和使用秩序，要求对服务的发布、使用、版本更新、下架和注销的生命周期各个环节进行规范化管理：

- 所有数据服务必须通过一个服务发布申请和审批流程才能发布到服务层，才能供服务消费者使用。数据服务权限分为数据权限与服务权限。
- 已经发布到服务层的数据服务可供服务消费者使用。服务消费者在使用某个数据服务时，必须首先申请该数据服务，获得该数据服务的使用权限。
- 服务层提供数据服务的目录浏览、搜索和详细信息查看功能。服务消费者可据此了解可供选择的数据服务，数据消费者可从中选择需要的数据服务。
- 对于免费使用的数据服务，服务消费者选择后立即具有该数据服务的使用权限。对于权限控制使用的数据服务，则需要根据使用协议申请获得一定期限的使用权限。
- 服务消费者通过调用 REST API 的方式使用数据服务，并将服务消费者账号、密码加密后作为参数传入 REST API 进行用户鉴权。
- 数据服务发布后，数据服务提供者可对自己发布的数据服务进行版本更新和基本信息修改。
- 可以根据需要对自己发布的数据服务进行下架或者注销，下架后的数据服务可以进行重新上架，下架后注销的服务无法重新上架。
- 为了加强服务在使用过程中的管控，每个数据服务必须按要求记录运行日志。服务层提供数据服务的运行日志查询功能，并统计每个数据服务的使用频次，供服务管理者对数据服务的使用情况进行审计。

7.6 应用域

应用域是运行在大数据平台上应用的集合。每一个应用都对应一个业务需求，实现一组特定的业务逻辑，并且通过服务接口与用户交互。应用域上的应用需要具有以下三个基本特征。

特征一：这些应用能够通过浏览器访问，或者具有开放的 API，允许用户或者客户端的调用。大数据平台的理想模式是不论用户身处何处，不论使用何种终端，只要有互联网连接和标准的浏览器，便可以不经任何配置地访问属于自己的应用。

特征二：应用域要求高度的整合，而且应用之间的整合兼容能力对于大数据平台的成功是至关重要的。应用之间的整合能力对于完美的用户体验来说是不可或缺的，因为用户的需求往往是综合性的。如果用户所需要的多个功能是由若干个彼此之间无法整合的应用程序来实现的，那么用户体验和操作效率都会大为降低。由于应用都是运行在平台中而且彼此相对独立，因此应用整合较传统应用相对容易实现。

特征三：用户在使用应用域时，一定要在功能上相互独立。各应用系统以服务的方式提供给用户，用户不必关心应用所在的运行环境，只需要提供应用所需输入数据即可得到软件的计算结果或报表。

应用域开发主要以界面为主，和传统的应用系统基本类似。有所不同的是，应用系统后端的服务器资源、存储资源、计算资源都由大数据平台统一提供，通过和服务层的接口对接，能使前端应用系统快速具备数据提取、计算和展示能力。由于大数据的高速发展，应用大数据的行业需求越来越多，同时也是各有不同，解决了后台的存储、计算能力，以互联网行业为代表的 DSP、推荐系统、用户画像等系统，制造业为代表的智能报表、智能生产、智能管控系统，金融业为代表的风险预测、信用评级等系统，都可以基于大数据平台进行快速开发上线。

7.7 本章小结

本章中的很多内容在之前已经介绍过了，因此这里没有着重就所有知识点详细讲解。当然，真正做大数据项目的开发设计时，概要设计、详细设计以及功能模块的设计要比本章介绍的复杂得多，其中的技术涉及本书中所有章节的内容，因此是一个更完整的方案。不同的公司在整合设计时需要根据自身特点、需求、背景、规划、限制和约束条件等综合考虑。

本章有以下内容需要读者重点掌握：**掌握每个模块常用的方法方式、工具以及各自的特点及适用场景。**

大数据 workflow

企业大数据的整体工作流程如何开展？不同环节间的工作关系和依赖条件是怎样的？各个大数据流程内部的工作内容分别是什么？本章将主要围绕这些内容展开探讨。

大数据工作流程从底层到上层依次是数据源、数据处理、数据存储、数据计算和数据应用，这五个环节相互依托不可分割；同时，作为对数据质量相关的工作标准和规范的管理，也贯穿了整个工作流过程。整个工作逻辑和功能如图 8-1 所示。



图 8-1 通用企业大数据功能逻辑图

8.1 数据源

数据源指的是企业内、外部数据的来源。数据源是大数据工作流的起点，一个完整的、需要不断迭代更新的数据工作都应该从数据源开始。这些数据源由于各自生产、存储环境的不同可分为日志/文件、数据库、网络爬虫、第三方 API/合作等。

8.1.1 日志/文件

在企业内部数据源中，会存在各种日志或文件类型的数据，可能包括以下几类：

- ❑ 日志数据：包括机器日志、用户访问日志、监控日志等，这些日志通常都是以半结构化的文本文件形式进行存储。
- ❑ 视频数据：主要是动态影像数据，例如宣传视频、操作视频、监控视频、医疗视频、记录视频等，这些大多是非结构化的数据并以文件的形式进行存储。
- ❑ 音频数据：主要集中在客服坐席录音领域，以非结构化文件的形式进行存储。
- ❑ 图片数据：主要集中在监控、生产、医疗、商品、服务等领域，这些数据是以非结构化文件的形式进行存储。
- ❑ 办公文件：主要集中在办公类系统如知识管理系统、办公系统、邮件系统、日常文档、项目信息等，以非结构化文件的形式进行存储，且文件类型、格式、内容等差异性较高。

日志和文件类数据是一类典型的半结构化或非结构化的数据源，这是大数据挖掘深度价值的宝贵资源。在大数据技术出现之前，企业往往无法处理这些信息并很难从中挖掘潜在知识。借助于大数据特有的技术，例如语音识别、图片识别等模式识别技术以及中/英文分词、自然语言处理、文本挖掘等相关技术，这些数据中的信息可以被提炼并应用到企业运营实践。

8.1.2 数据库

以数据库形式存储的结构化数据几乎是所有企业都具备的数据源，这些数据源往往包括企业生产和经营各个环节的核心数据，常见的数据库数据包括 CRM 数据、CC（呼叫中心）数据、财务数据、仓储数据、销售数据、物流数据、网站数据。

- ❑ CRM 数据：即企业客户管理系统相关数据。几乎所有的企业都有 CRM 相关数据，这是分析目标客户的关键。
- ❑ CC 数据：即企业呼叫中心系统相关的结构化数据，这些数据中主要包含记录和统计性的数据，例如呼叫记录、通话时长、接通等数据。
- ❑ 财务数据：包括现金流、资产管理、盈利、负债等数据，财务数据是企业数据的核心，也是成本结算的最终依据。任何业务系统的费用、考核、结算都应该以财务结果为准。
- ❑ 仓储数据：包括库存周转、库存结构、畅销、滞销等数据。仓储数据是传统品牌商和渠道商企业运转的关键枢纽。
- ❑ 销售数据：包括渠道、平台、品类等维度销售数据。销售数据是零售企业数据的核心。

□ 物流数据：包括出库、配送、调度、退换货等数据。

□ 网站数据：即流量数据，包括网站所有营销数据、用户数据、运营数据、在线销售等行
为日志。大多数企业的网站分析系统的数据结果都是以结构化的库表的形式进行存储。

在大数据相关技术出现之前，结构化的数据已经是企业内部数据工作的主要对象。大数据技术拓展了结构化数据处理过程中对于海量、实时性、低维护成本、灵活扩展性的需求，使得全量数据的处理成为可能。

8.1.3 网络爬虫

网络爬虫又称网页蜘蛛、网络机器人，它是按照一定的规则，自动抓取目标网站或全部网站信息的计算机程序或脚本。网络爬虫不是一个新生事物，在搜索引擎出现时网络爬虫就已经出现。

在大数据时代，很多企业可以自己编写脚本来爬取特定网站的信息，主要包括：

□ 商品信息：电子商务企业之间经常通过网络爬虫抓取竞争对手的商品信息，主要包括价格、库存、图片等信息，用来做自身动态商品定价的基础策略。另外，也可以通过抓取的商品信息了解对方的商品上下架、售卖、库存等信息，以此来辅助商品销售策略的制定。

□ 交易信息：对于交易性网站来讲，真实的交易数据往往都是无法直接获取的。但在网页上存在的类似于销量、预定量、预定人数等数据可以间接做交易信息的参考。

□ 用户信息：通过爬虫抓取的用户信息包括各个方面，主要包括用户属性、发帖、评论、咨询、关注、群组等，以此来获得用户真实行为和属性，并丰富企业内部用户数据。另外，抓取的匿名数据也可以作为基础用户调研或竞争对手研究的一部分。

□ 资讯信息：新闻资讯类网站间也经常抓取对方的新闻和资讯，并将这些信息发布到自身网站。另外，也可以通过企业发布的资讯来获得企业最新动态，辅助于竞争对手分析。

□ 竞争情报：企业可以利用当前流行的主题爬虫技术，建立网络竞争情报系统，系统能自动搜集指定竞争对手和指定领域内商品、服务、价格、渠道等实时信息，并能智能扩展或缩小收集范围，为战略战术研究和实时战术调整提供辅助决策作用。

□ 行业信息：很多企业也开发了爬虫技术来抓取行业机关、组织、政府等的全局观测信息，包括政策变化、市场动态、宏观形势、媒体舆论等。这对战略辅助决策起到一定的支持作用。

企业对于网络爬虫的信息获取大大提高了企业对于市场、竞争、情报、用户的信息获取、沟通和反馈效率，同时又能实时或准实时地监控整体反馈效果，对于企业的经营决策具有重要意义。

8.1.4 第三方 API/ 合作

很多企业也可以通过 API 或合作的形式获得更多的企业外部数据，这些数据由企业外部

产生，企业间通过合作、购买、交换等形式获得。企业外部数据通常包括竞争数据、营销数据、物流数据、行业数据等。

❑ 竞争数据：通常是关于竞争对手的流量、销售、产品、营销等方面的数据，例如竞争对手产品价格、竞争对手会员数据、营销投放渠道等。

❑ 营销数据：指企业通过营销或推广合作，获取自身站外相关媒体、渠道的曝光、点击、投放等详细数据。

❑ 物流数据：指第三方物流数据。

❑ 行业数据：指通过购买、调研等获得关于市场整体行情、市场趋势、用户结构、竞争环境等信息，常见于行业报告数据。

企业外部数据的获取正成为越来越多企业拥有更多数据的重要方式，并且市场上也出现了一些可以直接进行数据交易的组织和企业，而可进行交易的数据往往都是经过处理或脱敏的，基本都是匿名的，对于整体战略研究的重要性较大，但对于微观的经营决策影响较小。

在企业大数据工作的整体流程中，上述数据源通常都会通过一定的接口或组件进行统一集成，然后进入到数据处理层。

8.2 数据处理

在数据处理层，输入的是来自于不同源系统或环境的分散数据，输出的是具有一定关系和逻辑的高质量数据。数据处理层主要包括数据质量校验、清洗转换、质量提升、数据脱敏、集成整合等。

8.2.1 数据质量校验

数据质量校验是数据源进入数据处理环节的第一步，数据质量校验从完整性、一致性、及时性、有效性、准确性、真实性六个方面进行衡量。

1. 完整性

数据完整性包含数据值完整性和数据库完整性两方面。

(1) 数据值完整性

数据值完整性即数据值本身的完整程度，可分为两个层面：一是数据库内没有数据记录丢失；二是每一条数据记录内的相应属性均没有缺失值。

举例：

示例一：用户的每个订单动作都会触发订单记录，如果存在订单记录丢失，那么数据值不完整。

示例二：如表 8-1 所示，每条数据均有 3 个字段栏位（列），完整的数据值是三个栏位的值都是完整的；如果出现表中第二条记录缺失 PhoneNum 值的情况，那么数据值不完整。

表 8-1 数据值不完整性示例

UserID	Name	PhoneNum
12345	Lucy	18645678902
23456	Hanmeimei	
23599	Lilei	13453320191

注意 在数据质量校验过程中，数据字段的缺失校验比较容易发现，而数据记录的缺失审查和处理难度较大，原因是数据记录的缺失通常需要多表核对，在复杂数据场景中实现较为困难。

(2) 数据库完整性

数据库完整性是为了防止数据库接收不符合定义规则的数据而提出的，即数据库只能接收规则之内的数据。比如，数据库定义手机号码的字段只能是 11 位字符串，如果出现了位数不对则不予采集。

数据库完整性通过实体完整性、参照完整性、域完整性、用户自定义完整性四个方面进行约束。

- ❑ 实体完整性：实体即数据所要描述的客观主体，例如会员、订单等都是数据实体（常用 UserID、OrderID 表示），实体完整性要求主键不为空并且唯一。
- ❑ 参照完整性：参照即表之间的主键和外键关系，例如订单 ID 可能同时是会员表的外键、订单表的主键，通过订单 ID 可以关联会员表和订单表之间的关系。参照完整性要求不能引用不存在的主体。
- ❑ 域完整性：域完整是针对具体关系数据库的约束条件，反映了具体应用所涉及的数据必须满足的语义要求，例如数据类型、值范围、是否为空等。
- ❑ 用户自定义完整性：指用户自己定义必须满足的条件。

注意 数据库完整性是针对关系型数据库而言的，而对于 NoSQL 等非关系型数据库则基本不成立，原因是非关系型数据库强调面向集合、模式自由，因此不同行的数据约束条件可以完全不一样。

2. 一致性

与数据完整性类似，数据值一致性也包括数据值一致性和数据库一致性两方面。

(1) 数据值一致性

数据值一致性指不同库之间、不同表之间代表相同实体的数据值是相同的，以及同一个数据对象在不同用户访问时的值相同。

举例：

示例一：用户订单数据在数据表 A、B、C 中都有涉及，数据值一致性要求相同条件下（例如相同时间、相同维度）无论 A、B、C 是否处于相同的系统、库还是表，其结果要一致。

示例二：用户 User1 和 User2 在访问数据表 A 中的用户订单数据时的结果必须一致。

常见导致数据值不一致的原因包括四个方面：

- ❑ **数据更新延时问题。**多个数据系统、库、表之间更新都有一定时间的延迟，并且视数据量、更新频率、更新效率等其延迟时间不同，如果时间过长会导致数据不一致性的问题，这是最常见的数据不一致的场景。
- ❑ **请求并发控制问题。**多个用户同时访问一个数据对象时，由于并发控制不当而导致的数据不一致，主要对象是具有创建、编辑和删除权限的用户。
- ❑ **客观运行故障问题。**数据运行所依赖的软件、硬件、程序甚至不可抗力因素导致的数据损坏或数据丢失问题也会导致数据不一致。
- ❑ **数据抽取逻辑问题。**很多时候，不同系统对同一对象的抽取逻辑是不同的，由于逻辑不同导致的数据不一致也很常见。

（2）数据库一致性

数据库一致性指相同数据主体在不同数据系统、库、表的标识相同、约束条件一致以及数据库事务处理的一致。

举例：

示例一：用户表、订单表、配送单表都有关于订单的数据记录，如果三个表对订单主体的定义不一致会造成后期应用的混乱问题，通常会用统一变量名标识如 OrderID。

示例二：对于订单的生成操作，订单表和商品表会分别记录订单和商品数据，并且这两个表的事务处理是一致的，否则订单和订单商品信息无法同步和关联。

3. 及时性

数据及时性指数据满足业务应用需求的及时程度。以下是时间性较强的业务需求：

- ❑ 现在要看公司的实时销售数据；
- ❑ 今天要看最近 7 天的订单变化趋势；
- ❑ 今天要看全年的会员活跃度情况。

任何业务对数据的响应时间都是有要求的，不存在无限期的响应时间（如果存在，那么说明数据对它们已经失去价值）；但数据的处理又需要一定时间才能完成，尤其是数据深入挖掘和价值发现需要的时间更长。因此，业务的期望时间和数据处理时间会形成一对矛盾，即数据及时性矛盾。数据及时性的矛盾主要存在于两类场景：

- ❑ **实时数据。**实时数据的关键是“实时”，比较常见的“实时”要求是秒、分级别。实时需求会对数据工作的各个环节造成很大压力，但实时数据对业务的数据决策具有非常重要的作用，尤其是重大事件、营销活动、广告投放时，实时数据通过实时预警、分析等手段降低业务风险并尽可能保证企业利益最大化。

❑ **历史数据。**历史数据的关键是“历史”，历史数据的主要特点是数据量庞大，从数据库中短时间内提取大量数据是一种挑战。比如，订单表中每天的订单数据正常导出需要2小时，如果需要一个月的数据至少需要60个小时（在不考虑异常条件的前提下），再加上数据处理和挖掘的时间可能已经无法满足业务及时性的要求。

4. 有效性

数据有效性指数据是否符合应用需求以及满足需求的程度。数据有效性的内容包括数据格式、数据类型、数据值范围以及其他业务规则。

以用户留存率分析为例，以下两种情况说明数据有效性较低：

情况一：原始数据中没有用户启动时间戳的字段，因而无法进行留存率分析。

情况二：原始数据中的启动时间戳字段，75%的数据的值为空，这导致无法进行留存率分析。

5. 准确性

数据准确性指数据准确的程度。数据准确是数据工作的基本前提，同时也是保障数据结果可信的必要条件。

数据准确性主要受以下三个方面影响：

❑ **数据采集的准确性。**如果原始数据采集时数据就出现错误，那么后期很难纠正。

❑ **数据处理的准确性。**数据处理过程中会涉及多种方法，包括抽样规则、异常处理、处理顺序、挖掘模型选择、调参问题等，每个环节的处理不当都会影响数据准确性。

❑ **数据应用的准确性。**主要是数据指标、算法、应用场景、应用逻辑的准确理解和使用，例如用销售额除订单量无法得出客单价。



注意 在企业内部的数据工作中，各种主客观原因导致企业内部不同系统数据存在差异性。虽然这种差异性不会对数据结果方向造成重大影响，但仍然会影响企业内部对大数据工作价值的信任度；尤其是关键数据如订单、销售等的差异会导致企业领导层怀疑数据的可信度。因此，对于数据的准确性需要尽量保持统一。

6. 真实性

数据真实性指数据的来源和结果真实，并且其解读过程真实有效。数据的真实性可从三个方面进行衡量：

❑ **数据来源是否真实，**造假数据、捏造数据都违反真实性原则。

❑ **数据结果是否真实，**是否出现由于数据展现而造成的数据误解以及数据结果的错误。

❑ **数据解读的真实性，**是否在数据解读过程中夸大或忽略客观事实，形成明显的对数据现象、本质和落地建议的利益取向问题。比如，故意夸大某些部门在促销活动中的贡献。



注意 数据真实性是数据工作的底线，主观上的造假行为不仅关系到工作的结果问题，还关系到个人信誉及品质问题，这会对从业者整个职业发展造成非常恶劣的影响。

8.2.2 清洗转换

当第一步的数据质量校验完成后，针对有问题的数据要进行的是数据清洗和转换，另外还包括对正常数据的转换。数据清洗的主要作用包括纠正错误、删除重复项、统一规格、修正逻辑、转换构造和数据压缩。

1. 纠正错误

错误数据是数据源环境中经常出现的一类问题。数据错误的形式包括：

- ❑ **数据值错误**：数据直接是错误的，例如超过固定域集、超过极值、拼写错误、属性错误、源错误等。
- ❑ **数据类型错误**：数据的存储类型不符合实际情况，例如日期类型的以数值型存储，时间戳存为字符串等。
- ❑ **数据编码错误**：数据存储的编码错误，例如将 UTF-8 写成 UTF-80。
- ❑ **数据格式错误**：数据的存储格式问题，例如半角全角字符、中英文字符等。
- ❑ **数据异常错误**：例如数值数据输成全角数字字符、字符串数据后面有一个回车操作、日期越界、数据前后有不可见字符等。
- ❑ **依赖冲突**：某些数据字段间存在依赖关系，例如城市与邮政编码应该满足对应关系，但可能存在二者不匹配的问题。
- ❑ **多值错误**：大多数情况下，每个字段存储的是单个值，但也存在一个字段存储多个值的情况，其中有些可能是不符合实际业务规则的。

这类错误产生的原因是业务系统不够健全，尤其是在数据产生之初校验和入库阶段的规则不规范，导致在数据接收输入后没有进行判断或无法检测而直接写入后台数据库。

2. 删除重复项

由于各种原因，数据中可能存在重复记录或重复字段（列），对于这些重复项目（行和列）需要做去重处理。

对于重复项的判断，基本思想是“排序和合并”，先将数据库中的记录按一定规则排序，然后通过比较邻近记录是否相似来检测记录是否重复。这里面其实包含了两个操作，一是排序；二是计算相似度。

常见的排序算法：

- ❑ 插入排序；
- ❑ 冒泡排序；
- ❑ 选择排序；
- ❑ 快速排序；

□ 堆排序；

□ 归并排序；

□ 基数排序；

□ 希尔排序；

常见的判断相似度的算法：

□ 基本的字段匹配算法；

□ 标准化欧氏距离；

□ 汉明距离；

□ 夹角余弦；

□ 杰卡德距离；

□ 马氏距离；

□ 曼哈顿距离；

□ 闵可夫斯基距离；

□ 欧氏距离；

□ 切比雪夫距离；

□ 相关系数；

□ 信息熵。

对于重复的数据项，应尽量经过业务确认并进行整理提取出规则。在清洗转换阶段，对于重复数据项尽量不要轻易做出删除决策，尤其不能将重要的或有业务意义的数据过滤掉，校验和重复确认的工作必不可少。

3. 统一规格

由于数据源系统分散在各个业务线，不同业务线对于数据的要求、理解和规格不同，导致对于同一数据对象描述规格完全不同，因此在清洗过程中需要统一数据规格并将一致性的内容抽象出来。

数据字段的规则大致可以从以下几个方面进行统一：

□ **名称**：同一个数据对象的名称首先应该是一致的。比如，对于访问深度这个字段，可能的名称包括访问深度、人均页面浏览量、每访问 PV 数。

□ **类型**：同一个数据对象的数据类型必须统一，且表示方法一致。比如，普通日期的类型和时间戳的类型需要区分。

□ **单位**：对于数值型字段，单位需要统一。比如，万、十万、百万等单位度量。

□ **格式**：在同一类型下，不同的表示格式也会产生差异。比如，日期中的长日期、短日期、英文、中文、年月日制式和缩写等格式均不一样。

□ **长度**：同一字段长度必须一致。

□ **小数位数**：小数位数对于数值型字段尤为重要，尤其当数据量累积较大时会因为位数的不同而产生巨大偏差。

□ 计数方法：对于数值型等的千分位、科学计数法等计数方法的统一。

□ 缩写规则：对于常用字段的缩写，例如单位、姓名、日期、月份等的统一。又如，将周一表示为 Monday 还是 Mon 还是 M。

□ 值域：对于离散型和连续型的变量都应该根据业务规则进行统一的值域约束。

□ 约束：是否允许控制、唯一性、外键约束、主键等的统一。

统一数据规格的过程中，需要注意的一点是确认不同业务线带来数据的规格一致性，这需要业务部门的参与、讨论和确认，以明确不同体系数据的统一标准。

4. 修正逻辑

在多数据源的环境下，很可能存在数据异常或冲突的问题。

比如，不同的数据源对于订单数量的数据统计冲突问题，结果出现矛盾的记录。通常，这是由于不同系统对于同一个数据对象的统计逻辑不同而造成的，逻辑的不一致会直接导致结果的差异性；除了统计逻辑和口径的差异，也有因为源数据系统基于性能的考虑，放弃了外键约束，从而导致数据不一致的结果；另外，也存在极小的数据丢失的可能性，通常由于并发量和负载过高、服务器延迟甚至宕机等导致的数据采集的差异。

对于这类的数据矛盾，首先需要明确各个源系统的逻辑、条件、口径，然后定义一套符合各个系统采集逻辑的规则，并对异常源系统的采集逻辑进行修正。

某些情况下，也可能存在业务规则的错误导致的数据采集的错误，此时需要从源头纠正错误的采集逻辑，然后再进行数据清洗和转换。

5. 转换构造

数据变换是数据清理过程的重要步骤，是对数据进行的标准处理，几乎所有的数据处理过程都会涉及该步骤。数据转换常见的内容包括：数据类型转换、数据语义转换、数据值域转换、数据粒度转换、表/数据拆分、行列转换、数据离散化、数据标准化、提炼新字段、属性构造、数据压缩等。

（1）数据类型转换

当数据来自不同数据源时，不同类型的数据源数据类型不兼容可能导致系统报错。这时需要将不同数据源的数据类型统一转换为一种兼容的数据类型。

（2）数据语义转换

传统数据仓库中基于第三范式可能存在维度表、事实表等，此时在事实表中会有很多字段需要结合维度表才能进行语义上的解析。比如，假如字段 M 的业务含义是浏览器类型，其取值分别是 1/2/3/4/5，这 5 个数字如果不加转换则很难理解为业务语言，更无法在后期被解读和应用。

（3）数据粒度转换

业务系统一般存储的是明细数据，有些系统甚至存储的是基于时间戳的数据，而数据仓库中的数据是用来分析的，不需要非常明细的数据，一般情况下，会将业务系统数据按照数

据仓库中不同的粒度需求进行聚合。

(4) 表 / 数据拆分

某些字段可能存储多种数据信息,例如时间戳中包含了年、月、日、小时、分、秒等信息,有些规则中需要将其中部分或者全部时间属性进行拆分,以此来满足多粒度下的数据聚合需求。同样的,一个表内的多个字段,也可能存在表字段拆分的情况。

(5) 行列转换

某些情况下,表内的行列数据会需要进行转换(又称为转置),例如协同过滤的计算之前, `user` 和 `term` 之间的关系即互为行列并且可相互转换,可用来满足基于项目和基于用户的相似度推荐计算。

(6) 数据离散化

将连续取值的属性离散化成若干区间,来帮助消减一个连续属性的取值个数。比如,对于收入这个字段,为了便于做统计,根据业务经验可能分为几个不同的区间: $0 \sim 3000$ 、 $3001 \sim 5000$ 、 $5001 \sim 10\,000$ 、 $10\,001 \sim 30\,000$ 、大于 $30\,000$,或者在此基础上分别用 1、2、3、4、5 来表示。

(7) 数据标准化

不同字段间由于字段本身的业务含义不同,有些时候需要消除变量之间不同数量级造成的数值之间的悬殊差异。比如,将销售额进行离散化处理,以消除不同销售额之间由于量级关系导致的无法进行多列的复合计算。数据标准化过程还可以用来解决个别数值较高的属性对聚类结果的影响。

(8) 提炼新字段

很多情况下,需要基于业务规则提取新的字段,这些字段也称为复合字段。这些字段通常都是基于单一字段产生,但需要进行复合运算甚至运用复杂算法模型才能得到新的指标。

(9) 属性构造

在建模过程中,有些也会需要根据已有的属性集构造新的属性。比如,几乎所有的机器学习都会将样本分为训练集、测试集、验证集三类,那么数据集的分类(或者叫分区)就属于需要新构建的属性,用户做机器学习不同阶段的样本使用。



提示

在某些场景中,也存在一些特殊转换方法。比如,在机器学习中,有些值是离散型的数据但存在一定意义,例如最高学历这个字段中包含博士、研究生、大学、高中这4个值,某些算法不支持直接对文本进行计算,此时需要将学历这个字段进行转换。常见的方法是将值域集中的每个值拆解为一个字段,每个字段取值为 0 或 1 (布尔型或数值型)。这时,就会出现 4 个新的字段,对于一条记录来看(通常是一个人),其最高学历只能满足一个,例如字段博士为 1,那么其余的字段(研究生、大学、高中)则为 0。因此,这个过程实际上是将 1 个字段根据值域(4 个值的集合)拆解为 4 个字段。

6. 数据压缩

数据压缩是指在保持原有数据集的完整性和准确性，不丢失有用信息的前提下，按照一定的算法和方式对数据进行重新组织的一种技术方法。

对大规模的数据进行复杂的数据分析与数据计算通常需要耗费大量时间，所以在这之前需要进行数据的约减和压缩，减小数据规模，而且还可能面临交互式的数据挖掘，根据数据挖掘前后对比对数据进行信息反馈。这样在精简数据集上进行数据挖掘显然效率更高，并且挖掘出来的结果与使用原有数据集所获得的结果基本相同。

数据压缩的意义不止体现在数据计算过程中，还有利于减少存储空间，提高其传输、存储和处理效率，减少数据的冗余和存储的空间，这对于底层大数据平台具有非常重要的意义。

数据压缩有多种方式可供选择：

- ❑ **数据聚合**：将数据聚合后使用，例如如果汇总全部数据，那么基于更粗粒度的数据更加便利。
- ❑ **维度约减**：通过相关分析手动消除多余属性，使得参与计算的维度（字段）减少；也可以使用主成分分析、因子分析等进行维度聚合，得到的同样是更少的参与计算的数据维度。
- ❑ **数据块消减**：利用聚类或参数模型替代原有数据，这种方式常见于多个模型综合进行机器学习和数据挖掘。
- ❑ **数据压缩**：数据压缩包括无损压缩和有损压缩两种类型。数据压缩常用于磁盘文件、视频、音频、图像等。

8.2.3 质量提升

在数据清洗过程中，一个重要的环节是提升原有数据的质量，尤其是对残缺值、空值以及异常值的处理。

1. 补足残缺/空值

由于各种主客观原因，很多系统存在残缺数据，残缺数据包含行缺失、列缺失、字段缺失三种情况。行缺失指的是丢失了一整条数据记录，列缺失指的是丢失一整列数据，字段缺失指的是字段中的值为空值。其中，空值也分两种情况：

- ❑ **缺失值**。缺失值指的是数据原本是必须存在的，但实际上没有数据。比如，年龄这个字段每个人都会有，所以如果系统强制验证时不应该为空。
- ❑ **空值**。空值指的是实际存在可能为空的情况，所以空值不一定是数据问题。比如，身份证号这个字段，只有成人之后才有这个字符串，因此也可能存在非成人的用户，所以可能为空。

对于缺失值和空值的填充处理主要包含两种方式：一是手工填入可能的值；二是利用规则填充可能的值。某些缺失值可以从本数据源或其他数据源推导出来，这就可以用数据分布的状态和特征，例如众数、中位数、平均值、最大值、最小值填充，或者使用近邻

分析甚至更为复杂的概率估计代替缺失的值,从而达到填充的目的,某些情况下也可以直接以未知或 unknown 填充,这是一种先期不处理而等到后期业务处理数据时再处理的方法。

提示 对缺失数据进行填补后,填入的值可能不正确,数据可能会存在偏置,导致数据并不是十分可靠。除了用明显的可以确定的规则来填充值以外,基于已有属性来预测缺失值是一种流行的方法。假如性别字段部分记录为空,可以将性别字段作为目标变量进行建模分析,对完整样本建模后得出缺失数据性别为男、女的概率,然后进行填充。对于更多的不确定值的数据样本,如果不影响整体计算逻辑,建议先保持原样;如果会成为计算和建模噪声的数据,则可以采取上述方法进行处理,尽量使其在建模过程中的作用消减。

2. 丢弃数据/变量

对于异常数据,包括缺失值、空值、错误值、不完整的数据记录等,除了使用上面讲的方法进行清洗、转换、提升外,还有另外一种方法——丢弃。丢弃也是提升数据质量的一种方法。丢弃数据的类型包含两种:

- **整条删除**: 指的是删除含有缺失值的样本。某些情况下,由于各种原因可能存在大量的有某些字段缺失的数据记录,这会导致完整的数据很少,此时需要慎重使用。因此,这只适合关键变量缺失,或者含有无效值或缺失值的样本比重很小的情况。
- **变量删除**: 如果某一变量的无效值和缺失值很多,而且该变量对于所研究的问题不是特别重要,则可以考虑将该变量删除。这种做法减少了供分析用的变量数目,但没有改变样本量。

提示 数据丢弃或删除操作要慎重执行。一方面,被丢弃的数据很可能存在业务实际意义,而这些意义作为开发人员是不清楚的;另一方面,后期可能会需要针对异常数据进行处理,并成为重要的研究课题。比如,营销领域存在流量欺诈,电商领域存在黄牛订单,银行保险领域存在高风险业务,这些课题对应的底层数据可能都是异常数据。

8.2.4 数据脱敏

数据脱敏是指对某些敏感信息通过脱敏规则进行数据的变形,实现敏感、隐私数据的可靠性保护,数据脱敏又称为数据混淆、数据保密、数据消毒、数据扰频、数据匿名化等。

在数据工作流程中,内部很多环境和应用环节使用的都是真实的数据和信息。其中不乏很多敏感数据,例如个人信息、公司现金流、企业资产负债信息、关键合作伙伴、核心产品资料等,这些信息一旦发生泄露,不仅会给公司本身带来巨大损失,也会给服务客户以及合

作伙伴带来重大影响。因此，数据脱敏是很多公司数据处理过程中的必要步骤。

数据脱敏管理不仅是一个技术话题，它涉及经营流程规范、权限管理、加密和解密机制、关联业务系统管理、信息和资料流通管理等公司运营和安全管理方方面面。限于主题和篇幅，这里只阐述有关数据脱敏的技术课题。

数据脱敏包含以下几个步骤：

1. 了解脱敏需求

首先要确定脱敏需求，业务部门、外包单位、安全管理部门、监察部门等联合提出业务需求，技术开发和权限管理人员根据需求，确定数据脱敏的方式和环境，以及数据的用户管理策略等。

2. 确定使用环境

数据的使用环境大致分为四类：

- ❑ **开发环境**：主要用来做功能性开发使用，该环境下通常可以使用非敏感性数据。
- ❑ **测试环境**：主要用来做单元测试和系统集成测试，对于单元测试而言，使用非敏感数据即可；对于系统集成测试而言，必须使用脱敏的数据。
- ❑ **生产环境**：主要用于满足真实系统运行所需的功能，该环境下直接使用原始数据而不进行脱敏，因为生产数据是原始数据。但是，如果某些情况下需要访问生产环境数据，这时需要进行动态脱敏。
- ❑ **应用环境**：主要用来满足后期数据查询、分析、报表、交换等应用，该环境下需要使用脱敏的数据。

3. 定义敏感数据

定义敏感数据要对敏感度进行分级，主要参考依据是数据的来源、用途、价值、保存时间、泄露破坏影响、法律法规对数据保护的要求、访问维护和修改数据人员等。对于不同级别的敏感数据，要有不同的规则和管理规范进行对应。比如，大多数情况下，可以将数据敏感级别定义为绝密、高保密、保密、可公开四个级别，敏感程度由高到低。

不同行业、不同公司对于敏感数据的分级定义有所差异，以银行业中的部分敏感数据举例，常见的敏感数据包括：个人姓名、身份证号码、地址、电话号码、银行账号、邮箱地址、所属城市、邮编、密码类（例如查询密码、取款密码、登录密码、支付密码等）、组织机构名称、营业执照号码、银行账号、交易日期、交易金额等。

4. 确定敏感关联

确定敏感关联意味着需要明确敏感数据与其他表和数据的关联影响关系，这是脱敏工作中比较关键的部分。

通常情况下，某些敏感字段对其他字段或表可能产生影响，这个阶段要对脱敏数据的各种使用场景下的关联性进行分析，确认敏感数据脱敏后能保证系统开发、测试、生产和交付时的数据可用性，并明确敏感信息字段名称、字段类型、字段长度、赋值规范等内容。

比如,用户身份证号是一个敏感字段,基于身份证号可能会有衍生字段,包括生日、地域、年龄等;同时,身份证号还可能作为用户身份的唯一识别信息,用来做多表关联的主键。如果该字段经过脱敏,那么对应到计算逻辑或关联关系的其他字段也需要使用相同的脱敏规则,并确保能产生相同的脱敏结果。

5. 建立脱敏规则

脱敏规则通常可分为可恢复与不可恢复两类。

- 可恢复类规则是指脱敏后的数据可以通过一定的方式,可以恢复成原来的敏感数据,此类脱敏规则主要指各类加解密算法规则。
- 不可恢复类规则是指脱敏后的数据使用任何方式都不能被恢复。一般可分为替换算法和生成算法两大类。替换算法是将需要脱敏的部分使用定义好的字符或字符串替换,生成类算法则更复杂一些,要求脱敏后的数据符合逻辑规则,即“看起来很真实的假数据”。

常见的数据脱敏算法或规则如下:

- **替换**: 将需要脱敏的部分使用定义好的字符或字符串进行替换。比如,将地区山东替换为 SD,广东替换为 GD,这种方式下对内部人员可以完全保持信息完整性,但容易破解。
- **重排**: 将原来的数据字段按照一定的规则重新排序,例如将 ABCDE 替换为 EDCBA,这是一种类似于替换的规则,也比较容易被破解。
- **加密**: 加密是指通过加密算法和加密密钥将明文转变为密文,这种方式在密码中早有应用。常见的加密算法包括 MD5、RSA、DES、IDEA 等,安全程度取决于采用哪种加密算法,一般根据实际情况而定。
- **截断**: 数据截断是指只选取数据的一部分,舍弃其他信息来保证数据的模糊性,是比较常用的脱敏方法,但这种方式往往对之后的开发、测试、生产和应用都不够友好。
- **掩码**: 对原有数据进行隐藏(类似于密码),但保留了部分信息,保证数据原有长度不变,对信息持有者更易辨别,例如火车票上的身份证号码信息。
- **日期偏移取整**: 舍弃精度来保证原始数据的安全性,一般此种方法可以保护数据的时间分布密度,例如将日期数据由 20160620 13:31:12 取整为 20160620 13:00:00。



提示 一个好的数据脱敏过程必须具备以下特点:数据可用性、满足关联关系、符合业务规则、数据易用性、规则可定制性。

8.2.5 集成整合

在对数据进行校验、清洗转换、质量提升、脱敏之后,就需要进行数据的匹配和整合。

数据整合可以解决数据的分散性,避免信息孤岛;提高数据资源的利用率,重用性高;

底层数据结构更加透明，便于提供统一的数据接口；性能和扩展性更强，便于基于整体统筹的运维规划；数据管控和安全管理能力更强，降低了数据风险；还可以通过完整数据提供更好的辅助决策和数据驱动能力。

通常对于大型企业而言，匹配和整合都是基于一定主题域进行的，整合后的数据通常存储于数据集市和数据仓库中。在大数据平台中，除了传统结构化数据存储的形态外，还可以通过增加半结构化和非结构化的信息，丰富数据的内涵和外延。

比如，传统的以用户为主题的数据集市或数据仓库通常包含用户的属性、网站/APP/手机流量行为、订单销售行为、客服记录、咨询行为、评价行为、退换货行为、促销响应行为等。当半结构化和结构化的文件或数据进来之后，我们又可以从用户的客户坐席录音、站外获得社交媒体行为、用户晒图、拍摄视频、分享的内容和博客文章、地理位置信息等信息获得更多用户真实喜好和特性信息，由此可针对性地建立用户 360° 标签。

8.3 数据存储

企业内部的大数据平台上，可能存在传统关系型数据库和大数据存储平台并行的情况。

8.3.1 关系型数据库

传统的关系型数据库表现存储的两种主要方式是数据仓库和数据集市。这是在数据源基础上，通过数据处理对数据进行整合，形成供上层计算或业务使用的数据仓库及数据集市。数据仓库和数据集市是面向业务决策或上层数据应用，是一个面向主题的、集成的、相对稳定的、反映历史数据变化的数据集合。

- ❑ **面向主题的：**该集合通常面向特定主题，并围绕主题组织数据，例如会员主体、订单主题、营销主题等。
- ❑ **集成的。**该集合相对于分散的数据源，经过数据抽取、转换、加载以及各种数据处理和清洗动作得到一个集成的数据集合，它可以将数据的关系关联起来。
- ❑ **相对稳定的：**该集合中的数据相对稳定，不会涉及频繁的数据变化操作如删除、修改等，主要的数据操作是查询、检索、导出等。
- ❑ **反映历史变化的：**该集合中的数据是历史所有数据的集合，因此能够反映自数据记录以来企业的数据趋势，并通过定量数据反映结果。

数据仓库和数据集市从物理结构上看通常包括数据抽取、数据存储和数据访问三部分。

- ❑ **数据抽取：**通过特定方法与数据源关联，并经过 ETL（抽取、转换、加载）完成数据抽取和处理过程。
- ❑ **数据存储：**数据存储是核心，数据仓库的存储结构按照组织方式可分为星型、雪花型、复合型等，不同的数据仓库结构按照第一范式、第二范式，甚至第三范式的组织规则构建。

□ **数据访问**：最终目的是供上层应用，因此需要具备可访问性。数据仓库的主体包括上层数据计算需求及业务抽取数据，例如数据挖掘、数据计算、产品报表、数据驱动产品、辅助决策产品和临时需求等。



注意 在数据仓库结果设计时既要考虑到少数数据冗余、高数据可访问、简易数据计算和相关数据主题结构，又要兼顾业务在实际应用时的便捷性和可理解性。

8.3.2 分布式文件系统

用大数据平台（例如 Hadoop）除了可以存储结构化数据外，还可以实现对半结构化和非结构化数据的存储，这样大数据平台可同时满足结构化、半结构化和非结构化数据的存储和处理需求。

采用分布式文件系统存储的优势是：

- **可扩展性高**：Hadoop 是一个高度可扩展的存储平台，性能扩展时只需新增节点即可。
- **硬件成本低**：Hadoop 为企业用户提供了极具成本效益的存储解决方案，因为它可以基于廉价的服务器甚至 PC 机集群运行。
- **灵活性高**：Hadoop 能够使企业轻松访问到新的数据源，并可以分析不同类型的数据，从这些数据中产生价值，这意味着企业可以利用 Hadoop 的灵活性从社交媒体、电子邮件或点击流量等数据源获得宝贵的商业价值。
- **半结构化和非结构化处理**：Hadoop 拥有独特的存储方式，可以将所有数据以文件的形式进行存储，然后结合特定的计算和处理方式，可以对非半结构化和半结构的信息进行解析和提取。
- **容错能力强**：使用 Hadoop 的一个关键优势是其容错能力。当数据被发送到一个单独的节点，该数据也被复制到集群的其他节点上，这意味着在故障情况下，存在另 2 个副本（默认是存储 3 份）可供使用。

传统的关系型数据库和分布式文件系统是相辅相成的，各自拥有不可或缺的特定应用场景。

8.4 数据计算

数据计算模块是整个数据架构的关键部分，所有底层数据几乎都需要经过数据计算框架输出。数据计算层既要通过不同算法模块实现对常规统计、分析和海量机器学习挖掘需要，同时又要根据应用场景对于时效性的需求，及时提供时效性支撑。

8.4.1 三种数据计算时效性

数据计算按照计算结果输出的时间性可分为实时计算和离线计算，部分企业还会在实时

计算和离线计算之间加入临时计算。数据计算模块对于大多数中小企业来说没有必要单独拆分，原因是较小的数据体量和应用需求下，完全可以通过数据实时计算获得结果。数据计算模块只对大中型企业或具备海量数据处理需求的企业有存在意义。

1. 实时计算

实时计算通常基于实时性数据需求产生，实时性数据需求基于特定场景和规则，受动态数据集、时间周期、算法变化等因素综合影响。实时计算要求数据每次都是实时收集、实时计算、实时反馈、实时输出。实时计算的时间需求通常都是秒级甚至微秒级，Yahoo 的 S4、Twitter 的 Storm 都属于这一类。

实时计算的应用更多地侧重于在线服务。实时计算的常见应用场景包括：站外基于用户行为的实时广告投放的 RTB 和 DSP 系统、站内基于用户行为的个性化推荐系统、站内广告竞价系统、网站实时信息推送服务、公司实时监测和智能预警、站内搜索系统、机器故障实时告警等。

举个例子，假如站内推荐引擎需要针对用户实时浏览行为进行挖掘，并在用户下一次点击后实时推荐出用户可能喜欢的产品或内容。算法层可能包括回归、协同过滤、关联、神经网络等，数据层需要综合用户属性、历史行为、站内搜索行为、站内购物行为，以及上一次行为，大型网站数据运算量可能达到上亿条，推荐结果要在用户浏览下一个页面时进行体现。

2. 离线计算

离线计算相对于实时计算，区别在于时间窗口不需要实时性，同时由于离线计算有相对充裕的时间可以对全量数据进行运算挖掘，因此其数据结果相对实时计算更准确。离线计算一般是批量处理数据的过程，例如利用 Hadoop 的 MapReduce 属于离线计算类。

离线计算的数据处理时间通常是分钟或小时级，甚至可能是天；数据处理量通常在 TB、PB 级以上。

离线计算的应用场景包括：用户流失预警系统、基于用户购买的挽回系统、用户特征和规则提取系统、数据分析系统和产品报表、用户画像系统、渠道和用户价值系统等除实时计算外的数据挖掘都采用离线计算方式实现。

3. 临时计算

临时计算是介于实时计算和离线计算之间的一种计算方式，它既能保持数据的相对实时性，又能兼顾数据结果的准确性，它是针对实时计算和离线计算中间层需求的一种过渡性解决方案。临时计算的处理时间在秒到分钟，数据处理量在 GB 到 TB。

常见的临时计算如实时查询、Add-hoc 等这些实时查询返回的可能是各种各样的结果，因此无法通过事前的预计算得到并存储。在发生请求时，通常需要根据用户不同的输入条件进行响应，但实时性的要求没有那么高。比如，即使返回结果在 2 分钟内出来，也可以满足分析场景。常见的对实时性要求比较高的数据会存储到 HBase 或者内存数据库 Redis、MongoDB 等，基于 Impala 的 Hadoop 的实时查询也在一定程度上解决了 Hive 的实时性的效率问题。

8.4.2 结构化数据计算

本节分基础计算和基础分析两方面介绍结构化数据计算。

1. 基础计算

由于数据的基础计算大多在数据库中完成,因此本节的基础运算部分,如无特殊说明,均以 MySQL 作为数据库应用平台进行函数示例说明。

算术计算

算术计算是对数值型数据常用的基础运算,包括以下几种:

- ❑ 三角函数: 返回以弧度表示的正弦、余弦、正切、余切数,例如通过 `SIN(90)` 返回 90 的正弦值 0.89。
- ❑ 反三角函数: 返回正弦、余弦、正切等,例如通过 `ASIN(1)` 返回 1 的反正弦值 1.57。
- ❑ 对数运算: 返回表达式的指数值,例如通过 `LOG(10 100)` 得到结果 2。
- ❑ 平方根运算: 返回某个数据的平方根,例如通过 `SQT(100)` 得到 100 的平方根为 10。
- ❑ 幂运算: 返回表达式的幂值,例如通过 `POW(2,3)` 返回 2 的 3 次方为 8。
- ❑ 数据截断: 返回截取固定位数小数后的数字,例如通过 `TRUNCATE(120.3652)` 返回只保留 2 位小数的结果为 120.36。
- ❑ 取近似值: 取出大于表达式的最小数或小于表达式的最大数,例如通过 `ROUND(2.66183)` 取出 2.6618 的四舍五入最接近的整数,并保留 3 位小数,结果为 2.662。
- ❑ 取最大/小整数: 返回大于或小于某个数字的最大整数,例如通过 `CEILING(12.3)` 返回大于 12.3 的最大整数是 13。
- ❑ 取绝对值: 返回表达式的绝对值,例如通过 `ABS(-10)` 取出 10 的绝对值为 10。
- ❑ 求模计算: 对数据进行求模计算(返回余数),例如通过 `MOD(26,10)` 返回 26 的余数为 6。
- ❑ 进制转换: 将数据转换为二进制、八进制、十六进制等,例如通过 `BIN(12)` 将 12 转换为二进制数 1100。
- ❑ 随机数: 返回一个随机数,随机数可能通常处于一定的值域区间,例如 `RAND()` 返回一个 0 ~ 1 的随机数。

日期计算

日期处理是涉及时间计算过程中经常用到的计算内容,主要内容涉及对时间的获取、截取、变换、调整等。

- ❑ 返回当前时间信息: 返回当前系统状态中的时间信息(日期、时间),例如通过 `CURTIME()` 返回当前系统时间。
- ❑ 返回表达式的日期: 返回表达式中的年、月、周、日、小时、分、秒、毫秒等,当月几号、当周星期几等,例如通过 `DATE(NOW())` 返回当前系统时间中的日期部分。
- ❑ 调整日期: 返回经过增加或减少固定时间间隔的日期或日期时间值,例如通过 `DATE_ADD('2016-05-26', INTERVAL 2 DAY)` 来返回日期推移 2 天后的结果 2016-05-28。

❑ 日期间隔：返回 2 个日期的时间间隔，例如通过 DATEDIFF ('2016-05-01', '2016-05-26') 返回天数间隔为 17。

❑ 变换时间：在某些场景下，系统中原有的时间格式可能为 Linux 时间戳（典型场景为服务器日志），此时可能需要将其转化为标准日期和时间格式，或者进行二者的相关转换。

字符串处理

字符串的处理和计算常用于文本类或字符串类数据，常用方法如下：

❑ 大小写转换：对特定字段进行转换为大写或转换为小写，例如通过 UPPER ('test') 转换为大写的 TEST。

❑ 截取：字符串截取是常见的对字符串的操作，包括左截取、右截取、中间截取等，例如通过 MID ('ABCDE', 2, 3) 取出 BCD。

❑ 查找：查找某个字符出现的位置，例如通过 FIND_IN_SET ('E', 'A, B, C, D, E') 找到 E 出现在第 5 个位置。

❑ 替换：将字符串中的某些字符替换为目标字符（可字符匹配，也可指定匹配的位置），例如通过 REPLACE ('ABCDEFGH', 'AB', 'OO') 将 ABCDEFGH 中的 AB 替换为 OO。

❑ 颠倒排序：对字符串返回颠倒字符顺序后的值，例如通过 REVERSE ('ABCDE') 返回 EDCBA。

❑ 拼接：将多个字符串拼接为一个字符串或者以特定分隔符做分割的字符串，例如通过 CONCAT ('A', 'E', 'N') 将三个零散字符串拼接为整体，返回结果为 AEN。

❑ 去掉空格：去掉字符串左侧、右侧的空格，例如通过 TRIM ('ABCD') 将首尾的空格去掉，返回结果为 ABCD。

❑ 其他：返回字符的 ASCII 码值，返回字符串的长度或比特长度，对特定字符做转义，返回重复字符串、返回二进制数等。

格式化处理

格式化处理通常是在不同类型的数据之间进行转换，这样一方面将利于底层存储时的效率最优化；另一方面也是完成特定场景下数据计算和应用的必要条件。

❑ 日期格式化：日期和格式的表达方法转换是常用场景，例如通过 DATE_FORMAT (NOW(), '%b %d %Y %h:%i %p') 对当前系统时间按照表达式中的语法转换为 May 26 2016 03:25 PM。

❑ 时间格式化：该方法与上述方法类似，只是对象由日期变为时间，例如通过 SELECT TIME_FORMAT ('18:06:25', '%H %k %h %I %l') 将其中的时间拆分为 18 18 06 06 6。

❑ 数字格式化：对数字以逗号作为分隔符处理并保留特定位数的小数位数处理，例如通过 SELECT FORMAT (12562.6655, 2) 保留两位小数，结果为 12 562.67。

❑ 用数字表示 IP 地址：IP 地址是很多设备的重要识别标志，通过将 IP 地址转换为数字可便于进一步计算，例如通过 SELECT INET_ATON('192.168.0.1') 将 IP 192.168.0.1 转化为数字 3232235521。

加密处理

加密是数据脱敏的一种方法，常见的加密算法包括 MD5、RSA、DES、IDEA 等。

- ❑ MD5 (Message-Digest Algorithm 5)，严格来说它不算加密算法，只能说是摘要算法，原因是其无法进行揭秘，这就意味着加密的过程是一个原数据销毁和新数据生成的过程。它是一个不可逆的字符串变换算法，能将任意长度的“字节串”变换成一个 128bit 的大整数。MD5 的典型应用是数字指纹和数字签名，其次是加密和解密应用，例如操作系统密码登录。
- ❑ RSA：由 RSA 公司发明，是一个支持变长密钥的公共密钥算法，需要加密的文件块的长度也是可变的，它是一种非对称算法。它易于理解 and 操作，是既能用于数据加密也能用于数字签名的算法。由于计算复杂度是 RSA 的关键瓶颈，因此其一般来说只用于少量数据加密。
- ❑ DES (Data Encryption Standard)：是一种对称算法，既可用于加密又可用于解密。由于其数据加密标准速度较快，适用于加密大量数据的场合。DES 算法在 POS、ATM、磁卡及智能卡 (IC 卡)、加油站、高速公路收费站等领域被广泛应用。
- ❑ IDEA (International Data Encryption Algorithm) 国际数据加密算法，IDEA 是作为迭代的分组密码实现的，使用 128 位的密钥和 8 个循环，因此在一定时期内会提供相当强的安全性。由于该算法出现的时间不长，因此还没有经过长期的时差考验和技术可靠性验证。目前 IDEA 在工程中已有大量应用实例，安全套接字层 SSL (Secure Socket Layer) 也将 IDEA 包含在其加密算法库 SSLRef 中。

类型转换

数据类型转换是异构数据源常见的操作之一，常见的类型转换包括几种形式：

- ❑ 不同数据类型间的转换：根据计算需求，将数据类型进行转换，例如实数数据类型转换、其他数据类型与字符串间的转换等，包括字符串型、短整型、长整型、单精度浮点型、双精度浮点型、日期型、时间型等和字符串类型间的转换。
- ❑ 不同编码方式间的转换：ANSI 与 UNICODE 是两种不同的编码方式标准，在字符编码上有差异。ANSI 采用 8 位编码，只能表示 256 个字符，对应适用于英文字母类的表示；而 UNICODE 采用 16 位编码，可以容纳汉字等上万的字符。
- ❑ 不同位数类型间的转换：对消息的处理中我们经常需要将 8 位的 BYTE、16 位的 WORD、32 位的 DWORD、LRESULT、LPARAM 或 WPARAM 之间进行转换。

2. 基础分析

基础分析是对数据基础性的统计和分析工作，主要侧重点是对数据的分布、基本规律的统计和探索。

集中度分析

数据集中度是用来表示数据向某个方向或某个对象聚集的程度，常用来分析整体的集中情况。下面以表 8-2 为例说明常见的描述数据集中度的方法。

表 8-2 针对关键指标的预警信息

NAME	SALE	EDU
wang	2 500	本科
li	1 800	博士
zhao	2 600	研究生
song	4 700	研究生

- ❑ 汇总求和：该方法返回的是表达式的汇总值，例如通过 SUM (SALE) 得到所有人的销售收入总和为 11600。
- ❑ 计数：计数是用来返回满足某个条件的记录的数量，例如通过使用 COUNT 方法对 EDU=' 研究生 ' 的记录做计数，返回结果为 2。
- ❑ 平均值：平均值和汇总求和类似，只是计算的是算术平均值，例如通过 AVG (SALE) 可以得到平均销售额为 2900。
- ❑ 中位数：中位数是返回观测数据中间的一个数据，如果观测数据为偶数，那么返回的是中间 2 个数字的平均值作为中位数。例如本例中，表格内有 4 个记录，那么对于 SALE 列的中位数是 2550（此时中位数和均值的结果有明显差异）。
- ❑ 众数：众数是一组数据中出现次数最多的数值，有时众数可能不只是一个。

离散度分析

数据离散度是用来表示数据分散或离散趋势的度量，常见的描述离散度的方法有：

- ❑ 最大值：某个字段中的最大值，例如表 8-2 中的 SALE 最大值为 4700。
- ❑ 最小值：某个字段中的最小值，例如表 8-2 中的 SALE 最小值为 1800。
- ❑ 极差：最大值和最小值的差值，例如表 8-2 中的 SALE 极差为 2900。
- ❑ 方差：各个数据分别与其平均数之差的平方的和的平均数，例如表 8-2 中的 SALE 方差为 1 566 666.67（保留 2 位小数）。
- ❑ 标准差：标准差是方差的平方根，例如表 8-2 中的 SALE 标准差为 1251.67（保留 2 位小数）。

分布状态分析

分布状态用来描述数据分布规律中偏倚或分布形状的状态。

- ❑ 偏度：偏度是统计数据分布偏斜方向和程度的度量，是统计数据分布非对称程度的数字特征，例如表 8-2 中的 SALE 偏度为 1.50（保留 2 位小数）。
- ❑ 峰度：峰度用来描述分布形态的陡缓程度，也可以理解为对称状态尾部的厚度，例如表 8-2 中的 SALE 偏度为 2.79（保留 2 位小数）。
- ❑ 频数分布：频数分布是在分组的基础上，把总体的所有单位按组排列，形成总体中各个单位在各组间的分布。

假设检验

假设检验（Hypothesis Testing）是数理统计中常用的方法，它实现的是根据一定假设条

件由样本推断总体。假设检验的基本实现思路是：根据研究课题的需要对研究总体做某种假设，记作 H_0 ；选取合适的统计样本，其选取结果要使得在假设 H_0 成立时，其分布情况已知；然后由实测的样本，计算出统计量的值，并根据预先给定的显著性水平进行检验，做出拒绝或接受假设 H_0 的判断。常用的假设检验方法有卡方检验、 T 检验、 F 检验、 U 检验、 Z 检验等。

□ 卡方检验 (Chi-square test/Chi-Square Goodness-of-Fit Tes)，又称为 χ^2 检验。卡方检验是统计样本的实际观测值与理论推断值之间的偏离程度。实际观测值与理论推断值之间的偏离程度决定卡方值的大小：卡方值越大越不符合；反之则越趋于符合；若两个值完全相等时，卡方值就为 0，表明理论值完全符合。卡方检验属于非参数假设检验，适用于布尔型或二项分布数据，基于两个概率间的比较，早期用于生产企业的产品合格率等，在网站分析中可以用于目标转化率等有比率度量的比较分析。

□ T 检验 (T-Test) 是最常见的一种假设检验类型， T 检验属于参数假设检验，所以它适用的范围是数值型的数据。 T 检验需要总体样本符合正态分布特征，主要用于样本含量较小 (例如 $n < 30$)，总体标准差 σ 未知的数据。 T 检验广泛应用于医学统计等领域。

□ F 检验又叫方差齐性检验。在两种样本检验方差是否相等时，需要使用 F 检验做验证。因此，它是一种基础的检验方法。 F 检验法是由英国统计学家 Fisher 提出的，主要通过比较两组数据的方差 S^2 ，以确定它们的精密度是否有显著性差异。至于两组数据之间是否存在系统误差，则在进行 F 检验并确定它们的精密度没有显著性差异之后，再进行 T 检验。

□ U 检验是在大样本 ($n > 30$) 的情况下，检验随机变量的数学期望是否等于某一已知值的一种假设检验方法。 U 检验适用于样本量 n 较大且符合正态分布的情况。

□ Z 检验是一般用于大样本 (即样本容量大于 30) 平均值差异性检验的方法，比较两个平均数的差异是否显著。



提示 多种检验方法之间存在差异和联系，具体使用时需要参照以下因素进行具体选择：样本大小、样本分布状态、方差齐性、方差是否已知、变量个数等。

变异数分析

变异数分析用于两个及两个以上样本均数差别的显著性检验的统计分析方法，用来分析多个群体中的计量型数据，以便比较变异的意义和分析其来源。变异数分析又称为方差分析，全称是 Analysis of Variance，简写为 ANOVA。

变异数分析从观测变量的方差入手，研究诸多控制变量中哪些变量是对观测变量有显著影响的变量。它主要用于以下几个方面：①均数差别的显著性检验；②分离各有关因素并估

计其对总变异的作用；③分析因素间的交互作用；④方差齐性检验。

方差分析要求观察样本的分布符合正态分布或近似正态分布，并且各组数据之间的方差具有齐性。

提示 方差分析跟 F 检验在很大程度上具有混淆性，原因是方差分析的思路、过程、方法和假设过程都与 F 检验基本一致。实际上，F 检验可以看做是一种通用的统计学中的理论和方法（很多其他统计分析也会用到 F 检验）；而方差分析是在 F 检验基础上偏场景化和实例化的应用。

相关性分析

相关性分析指对多个具备相关关系的变量进行分析，从而衡量变量间的相关程度或密切程度。相关性可以应用到所有数据的分析过程中，任何事物之间都存在一定的相关联系。

相关性不等于因果联系，相关性和因果联系可以用一个案例来说明：

做商品促销活动，由于价格折扣低导致网站订单量大增，因此线下的配送订单量需求很大；这个过程中，由于订单配送量大及其他原因导致配送期间的破损率增加。在这个案例中，商品折扣低与破损率增加并不是因果关系，即不能说因为商品折扣低所以商品破损率增加，二者之间是相关关系，都与促销活动相关。

相关性结果是一个相关系数矩阵，从上到下或从左到右的解读结果相同。现以表 8-3 为例说明：“会话”列作为解读列，从上到下依次查找矩阵数据得到相关性结果，会话与收入相关性为 0.499、与加入购物车转化率相关性为 -0.193、与订单转化率相关性为 -0.257。该结果说明了会员与收入成正相关、与加入购物车转化率和订单转化率成负相关，且会员与收入相关性较大。

表 8-3 多指标相关性分析

	会话	收入	加入购物车转化率	订单转化率	
会话	Pearson 相关性	1	0.499	-0.193	-0.257
	显著性 (双侧)		0.171	0.618	0.504
	N	9	9	9	9
收入	Pearson 相关性	0.499	1	0.656	0.636
	显著性 (双侧)	0.171		0.055	0.065
	N	9	9	9	9
加入购物车转化率	Pearson 相关性	-0.193	0.656	1	0.954 ^①
	显著性 (双侧)	0.618	0.055		0.000
	N	9	9	9	9
订单转化率	Pearson 相关性	-0.257	0.636	0.954 ^①	1
	显著性 (双侧)	0.504	0.065	0.000	
	N	9	9	9	9

①在 0.01 水平 (双侧) 上显著相关。

8.4.3 半/非结构化数据计算

在传统的数据计算和挖掘过程中,更多的是针对结构化数据进行的。在大数据处理场景中,会面临各种半结构化和非结构化的数据计算,包括语音分析、文本分析、图像分析和视频分析等。

1. 语音分析

音频属于多媒体中的一种重要媒体,它是非结构化数据的一种。人耳能识别的音频信号的频率范围在 20Hz ~ 20kHz,其中语音分布在 300Hz ~ 4kHz。声音经过模拟设备记录或再生,成为模拟音频,再经数字化成为数字音频。

提示 完整的音频分析的范围很广,除了本节介绍的语音分析外,还包括自然音频分析(例如地震波分析)、机器音频分析(机器监控声音)等。但在目前的应用需求推动下,语音分析相对成熟且在商业环境中的应用应该较为广泛,因此本节重点介绍语音分析的知识点。但读者需要明确,语音分析并不是音频分析的全部,在人耳能识别的范围内还有大量非语音信息可供提炼,另外还包括更广阔的人耳不能识别的音频信息,例如地震波、次声波、超声波等,这些对于大数据的深入挖掘具有重要价值并且有特定的应用场景。

语音分析就是通过语音识别、语音解析、语音理解、场景控制等技术将非结构化的语音信息转换为结构化的数据,以实现对海量语音文件的知识挖掘和分析。语音分析的过程根据不同的处理任务而有所不同,核心计算过程包括语音源、语音信号预处理、语音识别、语音理解、语音压缩和结果输出几个部分,如图 8-2 所示。

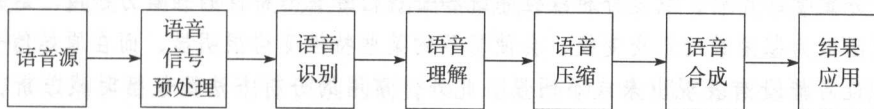


图 8-2 语音分析通用过程

语音源

在大数据平台中,语音的来源通常通过特定设备进行采集,或是以特定格式的语音文件存储于分布式文件系统内。常见的语音来源包括呼叫中心和客服中心的坐席录音、在线沟通工具的语音信息、智能应用程序采集到的语音指令、电信领域内的手机通话记录等。音频格式包括 pcm、amr、wav、voc、au、GSM、V3、vox 等。

语音信号预处理

语音信号预处理可分为两种方法:数字信号处理和模拟信号处理。目前对语音信号进行处理均采用数字处理,这是由于数字处理比模拟处理相比具有更多优点:

- 数字处理能够完成许多很复杂的信号处理工作，例如傅里叶变换、神经网络匹配等；
- 语音可以看出音素的集合，具有离散的性质，更适合于数字处理；
- 数字系统具有高可靠性、廉价、可扩展等优点，对于海量音频数据、实时数据需求的处理满足性高；
- 数字语音在数据传输和加密性等方面更加便捷、快速。

语音信号预处理阶段包含静音删除、声纹识别、噪声处理等过程，目的是去除环境噪声信息，并保留真实人声作为后期语音识别的有效语音。

- 静音删除：在语音文件中，通常都会存在一定的静音（静音中可能包含一定的噪声，也可能是完全静音），在语音识别之前需要将其中的静音删除。
- 声纹识别：语音处理的对象是人声，因此需要从音频信号中去除“非人声”的部分。如果是针对特定说话人，即特定语音对象的识别，还需要先甄别其他说话人的语音并自动删除。这个过程就是通过声纹识别并处理的。
- 噪声处理：在语音采集的过程中，不可避免地受到来自多方面的“噪声”干扰，具体包括各种环境噪声（例如汽车喇叭声、空调声、音乐等）、传输介质噪声、通信设备电噪声等。噪声会导致语音质量的下降，并使得语音处理、分析和挖掘的效率、效果和稳定性大打折扣。比如，通过在特定场景下采集到的声音进行处理得到的结果，在更换了采集环境后会导致语音识别结果变差。因此，在处理之前需要采用语音增强技术进行预处理，以减少“噪声”语音的影响。通常，“噪声”信号是随机发生的，因此要完全排除是不能的，噪声处理只能尽可能地降低噪声的干扰程度。



提示 在语音信号预处理过程中，会用到非常多的技术手段和方法，傅里叶分析在其中具有十分重要的作用，它是分析线性系统和平稳信号稳态特性的强有力手段。这种以复指数函数为基函数的正交变换，会使信号的某些特性变得很明显，而在原始信号中这些特性可能没有表现出来或不明显。此外，常用的分析方法还包括时域分析、频域分析、失真分析、频谱分析等。

语音识别

语音识别阶段是将语音进行信号建模处理，得到不同人声角色对应的语音信息。通过让机器识别和理解，把人类的语音信号转变为相应的文本或命令的技术。语音识别可分为角色识别和语音识别两步。

步骤1 角色识别

在语音预处理阶段的声纹识别，可能只保留特定角色或说话人主体。假如音频中保留了超过2个的说话主体，那么就需要针对不同角色分别进行语音识别，提取不同角色的声纹。比如，在坐席录音中，常见的两类角色是客服和终端用户两种。

角色识别过程的两个关键任务是特征提取和模式匹配。

- 特征提取：特征提取的任务是从说话人的语音信号中提取出说话人的声学特征，这要求说话人的声纹具有区分性强、稳定度高等特性，即说话人的声纹特征是个性的，且相对稳定的。
- 模式匹配：模式匹配的任务是将说话人的特征匹配到说话人主体上，输入特定特征后能匹配到特定的对象。提取到的特征信号通过算法训练后，建立起对应的声学模型和语言模型，为语音特征信号的匹配提供依据。



提示 角色识别解决的问题只是哪些语音信息对应的是哪些人的问题，对语音中的具体信息还没有做完整分析和解释。整个逻辑是确定多种声纹特征作为输入，输出的目标变量是角色或说话人。

步骤2 语音识别

在语音识别阶段，会将语音中的完整信息进行识别和辨认。语音识别过程主要包括选取语音识别单元、提取语音特征参数、模型训练、模型匹配几个阶段。

(1) 选取语音识别单元

选择识别单元是语音识别研究的第一步。语音识别单元有单词、音节和音素三种，具体选择哪一种，由具体的研究任务决定。

□ 单词广泛应用于中小词汇语音识别系统，不适合大词汇系统，原因在于模型库太大，训练模型任务繁重，模型匹配算法复杂，难以满足实时性要求。

□ 音节多见于汉语语音识别，主要因为汉语是单音节结构的语言，而英语是多音节，并且汉语虽然有大约 1300 个音节，但若不考虑声调，约有 408 个无调音节，数量相对较少。因此，对于中、大词汇量汉语语音识别系统来说，以音节为识别单元基本是可行的。

□ 音素以前多见于英语语音识别的研究中，但目前中、大词汇量汉语语音识别系统也在越来越多地采用。原因在于汉语音节仅由声母（包括零声母有 22 个）和韵母（共有 28 个）构成，且声、韵母声学特性相差很大。实际应用中常把声母依后续韵母的不同而构成细化声母，这样虽然增加了模型数目，但提高了易混淆音节的区分能力。由于协同发音的影响，音素单元不稳定，所以如何获得稳定的音素单元，还有待研究。

(2) 提取语音特征参数

特征提取就是完成从语音中提取出对语音识别有用的信息的过程。它对语音信号进行分析处理，去除对语音识别无关紧要的冗余信息，获得影响语音识别的重要信息。对于非特定角色的语音识别来讲，希望特征参数尽可能多地反映语义信息，尽量减少说话人的个人信息（对特定角色语音识别来讲，则相反）。从信息论角度讲，这是信息压缩的过程。特征提取主

要有以下几种方法：

- 线性预测 (LP) 分析技术是目前应用广泛的特征参数提取技术，许多成功的应用系统都采用基于 LP 技术提取的倒谱参数。但线性预测模型是纯数学模型，没有考虑人类听觉系统对语音的处理特点。
- Mel 参数和基于感知线性预测 (PLP) 分析提取的感知线性预测倒谱，在一定程度上模拟了人耳对语音的处理特点，应用了人耳听觉感知方面的一些研究成果。实验证明，采用这种技术，语音识别系统的性能有一定提高。
- CEP 利用同态处理方法，对语音信号求离散傅里叶变换 DFT 后取对数，再求反变换 IDFT 就可得到倒谱系数。对 LPC 倒谱 (LPCCEP)，在获得滤波器的线性预测系数后，可以用一个递推公式计算得出。实验表明，使用倒谱可以提高特征参数的稳定性。
- MFCC 先用 FFT 将时域信号转化成频域，之后对其对数能量谱用依照 Mel 刻度分布的三角滤波器组进行卷积，最后对各个滤波器的输出构成的向量进行离散余弦变换 DCT，取前 N 个系数。PLP 仍用德宾法去计算 LPC 参数，但在计算自相关参数时用的也是对听觉激励的对数能量谱进行 DCT 的方法。

(3) 模型训练及匹配

模型训练是指按照一定的准则，从大量已知模式中获取表征该模式本质特征的模型参数，而模型匹配则是根据一定准则，使未知模式与模型库中的某一个模型获得最佳匹配。语音识别所应用的匹配和训练技术主要有动态时间规整技术 (DTW)、矢量量化技术 (VQ)、隐马尔可夫模型 (HMM)、基于段长分布的非齐次隐马尔可夫模型 (DDBHMM) 和人工神经网络 (ANN)。

- 动态时间规整技术 (Dynamic Time Warping, DTW) 是较早的方法，它应用动态规划方法成功解决了语音信号特征参数序列比较时时长不等的难题，在孤立词语音识别中获得了良好性能。但因其不适合连续语音大词汇量语音识别系统，目前已被 HMM 模型和 ANN 替代。
- 矢量量化技术 (Vector Quantization, VQ) 是一种基于块编码规则的有损数据压缩方法。基于训练序列的 VQ 设计算法，对训练序列的运用绕开了多维积分的求解，产生了一种经典的被世人称为 LBG-VQ 的算法。
- 隐马尔可夫模型 (Hidden Markov Model, HMM) 是语音信号时变特征的有参表示法。它由相互关联的两个随机过程共同描述信号的统计特性，其中一个 is 隐蔽的 (不可观测的) 具有有限状态的 Markov 链，另一个是与 Markov 链的每一状态相关联的观察矢量的随机过程 (可观测的)。HMM 模型的训练和识别都已研究出有效的算法，并不断被完善，以增强 HMM 模型的鲁棒性。
- 基于段长分布的非齐次隐马尔可夫模型 (Duration Distribution Based Hidden Markov Model, DDBHMM) 是一个非齐次的 HMM 语音识别模型，从非平稳的角度考虑问题，

用状态的段长分布函数替代了齐次 HMM 中的状态转移矩阵,彻底抛弃了“平稳的假设”,使模型成为一种基于状态段长分布的隐含 Markov 模型。它比 HMM 模型具有更低的计算复杂度,并且由于解除了对语音信号状态的齐次性和对语音特征的非相关性的限制,因此为语音识别研究的深入发展提供了一个和谐的框架。

□ 人工神经网络 (Artificial Neural Network, ANN) 是语音识别应用的热点。ANN 是由大量处理单元互联组成的非线性、自适应信息处理系统,具有自适应性、并行性、稳健性、容错性、联想对比和推理演绎能力,但 ANN 不具有 HMM 模型的动态时间归正性能。因此,较好的方式是把 ANN 和 HMM 结合起来使用,以此来提高语音识别的准确性、适应性和鲁棒性等特征。

语音理解

在语音识别的基础上,需要对语义特征进行分析,目的是通过计算得到语音对应的潜在知识或意图,然后提供对应的响应内容或方法。语音识别和语音理解的差异之处在于,语音识别重在确定语音表达的字面含义,属于表层意义;而语音理解重在挖掘语音的背后含义,属于深层意义。语音理解的本质是了解语音背后的背景、含义、需求和目的,因此这是一种社会属性的行为。

语音理解基于语音识别的结果,还需要结合语言语法、语音学、自然语音处理、机器学习、知识搜索、知识处理等交叉学科和相关领域。语音理解需要解决几类问题:排除各种噪声信息、通过上下文理解语音含义、纠正不合理的语法或句子、结合社会属性处理信息。

但目前的实际情况是,语音理解对于固定模式和知识的学习可以得到较高的准确率,但对于具有社会属性和背景环境相关的语音理解目前的识别率较低。原因是很多语音是在特定的背景下产生,如果单纯看字面意思是无法准确判断语音的潜在含义,例如:

□ “你真行”。在表扬和批评的环境下都可以产生,此时需要有上下文关联。

□ “你造吗”。这个句子产生在网络时代,是网络流行语的产物,要使计算机能识别这个词,需要在网络时代的大环境下更新特定语料库。

□ “我差点没摔下去”。这个句子要表达的结果可能是没摔下去,但通用语法中双重否定即肯定的规则会让这个句子变为摔下去了,这是典型的社会语境下的潜在含义。

□ “只有我们 3 个人知道的事情他 (第四个人) 怎么会知道”。这句话的字面含义是询问“他”如何知道这件事情,但潜在表达的本意可能是,第三个人 (说话者和聆听者之外知道这个事情的人) 是个不太可靠的人。这属于潜在含义的表达。

□ “剩女的原因:一是谁都看不上,二是谁都看不上”。对于这个句子,如果通过机器来识别,那么对于两个原因可能“认为相同”,但人类却可以轻易地分辨,这是社会行为的价值。

□ “冬天:能穿多少穿多少;夏天:能穿多少穿多少”。这句话中对冬夏两季的解读,机器很难区分,但人类在结合社会穿着潮流的情况下就很容易理解。

目前，在机器理解的训练过程中，通常基于部分语音样本进行训练。语音训练会通过不同的时长（例如 3000 小时、8000 小时等）来作为产出效果的基本要求。在训练过程中，主要结合的技术方式是自然语音处理、神经网络、机器学习等。

综合来看，要实现机器对于语音的完全理解，需要机器跟人类一样“处于”社会生活中，在目前的条件下，暂时无法实现；另一条变通思路是将人类在社会群体生活中的相关语音信号，全部交给机器去训练，以此来达到“理解”的目的。但这种方法又面临着另外的问题——人类每天的社会群体生活语言表达的时间非常长，到底需要多少数据才算“足够”？这个标准很难确定；另外一个更严峻的问题，语音在人类群体生活中只是传递信息的一种方式，它仅仅是针对听觉产生的，很多时候我们可以通过视觉、触觉、味觉、嗅觉等方法传递信息，而这些是单纯语音无法获取的信息，因此要理解完整的信息又需要其他的信息源和数据分析方式。

语音压缩

语音压缩指的是对原始数字音频信号流运用适当的数字信号处理技术，在不损失有用信息量或极少损失信息量的前提下，压缩信号编码速率，以生成适合传输的数字信号流，这也称为压缩编码。这样做的优点在于可以适应在低码率的信道上实现可靠传输，也可以在同样的信道上传输更多的数据，提高传输效率的同时又能节约传输成本。

语音合成

语音合成就是让计算机能够“开口说话”，这是一种拟人的技术方法。语音合成，又称文本转语音（Text to Speech）技术，它通过机械的、电子的方法将文字信息转变为人类可以听得懂的语音。

语音合成过程是先将文字序列转换成音韵序列，再由系统根据音韵序列生成语音波形。第一步涉及语言学处理，例如分词、字音转换等，以及一整套有效的韵律控制规则；第二步需要先进的语音合成技术，能按要求实时合成出高质量的语音流。通常，文语转换系统都需要一套复杂的文字序列到音素序列的转换程序，它不仅要应用数字信号处理技术，而且必须有大量的语言学知识的支持。

优秀的语音合成可以做到多行业、多音色、多语种、多方言的综合场景覆盖，甚至可以通过对特定人的语音输出来定制专属语音服务（例如高德导航的语音导航）。另外，通过对音量、音高、语速等参数进行调节，对声场混响、回声、合唱、忽远忽近、机器人、背景音变换等可实现更复杂的场景应用需求。

结果应用


经过上述步骤之后，语音可以对外输出，输出的方式包括文本输出、语音输出、指令输出等。

□ 文本输出：语音识别完成后，直接通过终端输出文本信息，常用于语音转文字的应用。

比如，微信中的语音转文字属于该类应用。


□ 语音输出：语音输出包括原始语音、压缩后的语音、合成后的语音等形式，在设备终端通过一定的设备进行解析并播放。比如，QQ 语音属于输出压缩后的语音。

□ 指令输出：指令输出是在语音识别和理解的基础上，通过跟应用程序或系统结合，将语音理解后的文字转换成操作指令进行输出。比如，苹果的 Siri 属于此类功能，通过跟 Siri 的“沟通”，除了可以进行日常沟通，它还可以告诉你天气情况、帮你设置系统日程、介绍餐厅等。这是智能机器人在模式识别方面的典型应用。

 提示 在实际进行语音大数据平台搭建和运行过程中，除了上述流程和操作外，还需要特定语料库来分别进行不同场景的训练，例如标准普通话库、闽南语库、日语库、英语库等。另外，也需要针对不同的应用场景提炼特定的场景引擎，用来满足特定场景下不同语音的训练，例如电信行业、金融行业、旅游行业等。

2. 文本分析

文本分析就是通过对文本特征的挖掘和提取，找到潜在的意图、目标、特征和知识。文本分析是目前比较成熟的一类非结构化数据的应用，本节所阐述的文本分析内容涉及自然语言处理和文本挖掘。文本分析核心计算环节包括文本源、文本预处理、文本转向量、文本特征提取、文本挖掘应用几个部分。

 提示 在文本类分析中，很多读者往往很难准确分辨文本挖掘和自然语言处理之间的关系。实际上，要完成完整的自然语言的分析和挖掘功能，往往需要二者结合才能实现。但在特定的场景中，二者又有应用差异性和应用独立性：自然语言处理侧重于对自然语言的处理和分析，包括分词、词性标注、文本分类、自动提取标签等，另外也包含非常强的语言理解、语义解析等内容；而文本挖掘侧重于通过更深入的算法进一步挖掘文本价值，虽然当前的文本挖掘也是针对自然语言进行的，但其本身的适用范围可以超出自然语言的范畴，例如机器日志不属于自然语言但也可以做文本挖掘等。自然语言处理是自然语言的基础处理，文本挖掘则是拓展应用。因此，具体用到哪类方法需要视应用需求和场景而定。本节所提到的文本分析是综合二者的内容提炼的通用方法和流程，围绕的核心是自然语言的文本，而不包括其他类型的文本。

文本源

用于文本分析和挖掘数据的文本源可能具有不同的类型，并且分散在不同的源环境。常见的文本数据源包括：

- 数据库表/视图：存储于结构化数据库中的库、表、视图等，可以包括传统数据库中的文本类存储，例如 blog、text 等字段存储的文本数据；也可以是专用的文本数据库，例如 txtSQL、Txt DB API、Text Database 等存储的数据。
- 文档：很多情况下，文本也会存储于一些文档或平面文件中，例如 word 文档、txt 记录等。

- 网页：网页是文本源非常集中的一类来源，而且很多文本信息都是通过爬虫方式抓取得到的，特定信息包括网站情报、用户评论信息、社会化媒体行为等。
- 邮件：邮件是通信中的重要文本源，每个公司几乎都有自己的专有邮件系统，其中包含垃圾邮件、内部通信信息和外部沟通信息等。
- API：不同系统间的文本交换，或跟外部的数据交换通常通过 API 实现，其中包含文本类信息源，例如文献库、新闻库、微博 API 等。
- 语音识别：在语音识别和解析之后，形成的文本信息可供文本分析和挖掘使用。
- 其他：包括书籍、文章等。

文本预处理

与数据库中的结构化数据相比，文本具有有限的结构，某些类型的数据源甚至没有数据结构。因此，预处理就是要对半结构化或非结构化的文本进行格式和结构的转换、分解和预处理等，以得到能够用于进一步处理的基础文本。预处理的方式包括半结构化转结构化、基础文本处理、分词、词性标注、词频统计等。

(1) 半结构化转结构化

将原始半结构化或非结构化的语料转换为同一类型的结构化数据，便于后续的统一处理。比如，在微博的发帖数据，将其中的微博 ID、发帖时间、发帖来源、微博内容、表情、图片、视频、话题、转发数、评论数、点赞数、@ 微博 ID 等信息提炼出来。

(2) 基础文本处理

根据不同的文本数据来源，可能涉及的基本文本处理包括去除无效标签、编码转换、文档切分、基本纠错、去除空白、大小写统一、去标点符号、去停用词、保留特殊字符等。

- 去除无效标签：比如，从网页源代码获取的文本信息中包含 HTML 标签，此时要提取特定标签内容并去掉标签。
- 编码转换：编码不同对于中文处理具有较大影响，例如 UTF-8、UTF-16、GBK、GB2312 等之间的转换。
- 文档切分：如果获得的单个文档中包含多个文件，此时需要进行单独切分以将不同的文档拆分出来。
- 基本纠错：对文本中明显的人名、地名等常用语和特定场景用语的错误进行纠正。
- 去除空白：文本中可能包含的大量空格、空行等需要去除。
- 大小写统一：将文本中的英文统一为大写或小写。
- 去标点符号：去除句子中的标点符号、特殊符号等。
- 去停用词：常见的停用词包括 the、a、an、and、this、those、over、under、above、on 等。
- 保留特殊字符：某些场景下可能只需要对汉字、英文或数字进行处理，其他字符都需要过滤掉。

(3) 分词

分词是将一系列连续的字符串按照一定逻辑分割成单独的词。在英文中，单词之间是以

空格作为自然分界符的；而中文只有字、句和段能通过明显的分界符来简单划界，而作为词是没有形式上的分界符。因此，中文分词要比英语等语种分词困难和复杂得多。对于复杂的中文分词而言，常用的分词方法包括最大匹配法、逆向最大匹配法、双向匹配法、最佳匹配法、联想－回溯法等。

(4) 词性标注

词性标注是将句子中带有兼容词性的单词根据上下文确定出唯一的词性含义。在不同的语言中，都存在大量词的兼容性问题，因此如何排除歧义是词性标注的关键问题。常见的兼容词分类包括：

- 同型异性异义词：例如领导包括动词和名词两种；
- 同型异性同义词：例如小时可以作为量词和名词；
- 异型同性同义词：例如电脑和计算机属于同性同义词。

(5) 词频统计

词频统计是对每个单词出现频次和频率的统计。英文词频的统计由于都是基于空格分词，相比于中文统计会简单很多；而中文词频统计由于涉及语义及分词的不同，词频统计的难度更大一些。在词频统计之后，通常都会进行归一化以减少数据规模对词频的影响。

词频统计也是很多文本分析挖掘的基础模块，例如文本分类、情感分析、用户模式分析、非结构化信息提取（例如关键字、文摘）等过程中也会用到词频统计。常用的词频统计算法包括：TF-IDF、KMP、SVM 等。

文本转向量

目前，人们通常采用向量空间模型来描述文本向量，即将文档作为行，将分词后得到的单词（单词会在向量空间模型中被称为向量，也被称为特征、维度或维）作为列，而矩阵的值则是通过词频统计算法得到的值。这种空间向量模型也被称为文档特征矩阵。其表示方法如表 8-4 所示。

表 8-4 空间向量模型示例

	特征 1	特征 2	特征 3
文档 1	0.015	0.043	0.018
文档 2	0.013	0.081	0.092
文档 3	0.079	0.017	0.018

文本特征提取

文本特征提取是文本分析过程中非常重要的一步，但不是每个场景都必须使用的。通常情况下，如果要对文本按维度进行挖掘，例如进行聚类、分类等操作，输入的维度如果不加归约和选择，那么将面临维度灾难——这种未经处理的特征不仅会增加后续工作的计算资源，也会使整个处理过程的效率非常低下，更重要的是可能会影响分类、聚类等算法结果的精确性。

文本特征提取就是要找到能代表文档特征的单词或单词的聚合，然后针对这些特征进行

后续挖掘和计算。通常，被选择的特征能代表文档的概率越高，证明特征越好。

对于特征的提取，常用的方法如下：

- ❑ 用映射或变换的方法把原始特征变换为较少的新特征，常见的 PCA 和 LDA 就是映射原理。
- ❑ 从原始特征中人工挑选出一些最具代表性的特征，这也是一种人工的方法，但往往都是根据数据的实际分布并结合专家经验或操作者经验进行选择。
- ❑ 用数学的方法进行选取，找出最具分类信息的特征，这种方法是一种比较精确的方法，人为因素的干扰较少，尤其适合于文本自动分类挖掘系统的应用。

文本挖掘应用

文本挖掘和应用是最终文本分析的落脚点，常见的挖掘和应用包括文本聚类、文本分类、非结构化信息抽取、文本纠错、相关文本推荐、文档相似度判别、情感分析等。

(1) 文本聚类

文本聚类就是要在一大堆文档中找到哪些文档具有较高的相似性，然后可以针对这些相似性文档的聚合进行类别划分。

文本聚类应用场景：提供大规模文档集进行类别划分并提取公共内容的概括和总览；找到潜在的各个文档间的相似度进行相似度判别、类别修正，以减少浏览相似文档和信息的时间和精力。

文本聚类常用方法：层次聚类法、平面划分法、简单贝叶斯聚类法、分级聚类法、基于概念的文本聚类、混合模型聚类、光谱聚类、潜在语义标引聚类 (LSI) 等。

(2) 文本分类

文本分类也是将文本划分为不同的类别，与文本聚类的区别在于文本聚类没有 Label 可用于训练，因此它是一种非监督式的学习；而文本分类有特定的 Label 可供学习和训练，这是一种监督式的学习方法。从实际应用角度，聚类提供的是在没有任何经验或先前知识的前提下，对大规模文本进行类别自动划分；分类提供的是基于已有的训练模式和 Label 属性，预测其类别所属。

文本分类应用场景：信息的类别划分，例如将网页的资讯自动划分为影视、音乐、健康、财经、汽车、政治等类别，将根据邮件内容进行垃圾邮件过滤，针对论坛、博客等社会化媒体中恶意帖子的识别和过滤等。

文本分类常用方法：朴素贝叶斯、矩阵变换法、K-近邻、支持向量机、神经网络等。

(3) 非结构化信息抽取

非结构化信息抽取指的是从文本提取特定非结构化信息，包括摘要、关键字等。非结构化信息抽取能生成简短的关于文档内容的指示性信息，将文档的主要内容或核心关键字呈现给用户，以决定是否要阅读文档的原文，这样能够节省大量的浏览时间并提高关键信息的展示能力。

非结构化信息抽取应用场景：帖子、新闻、资讯、评论、问答等。

非结构化信息抽取常用方法：通过词频统计获得文本的主要关键字，而摘要提取方法包

括自动摘录、基于理解的自动文摘、信息抽取和基于结构的自动文摘等。

(4) 文本纠错

文本纠错能够实现对文本的自动纠错功能,这是一种辅助输入的功能。文本纠错包含字词级别的短文本纠错、语法搭配纠错和长句子的语义纠错,目前主要的文本纠错侧重于短文本纠错。

文本纠错应用场景:文本编辑器纠错、搜索引擎输入内容纠错、输入法的输入纠错、书籍和稿件校正等。

文本纠错常用方法:基于机器学习算法的纠错方法是较为广泛且精确的自动纠错方法,包括 SVM、贝叶斯、神经网络、逻辑回归、决策树等,除此之外也有通过特定模式和规则的泛化匹配、N-Gram 模型判断文本中的错误字词。

(5) 相关文本推荐

用户在某些文本之间可能存在频繁的关联查阅关系,而这些关系之间会蕴藏用户的潜在意图,这可以通过相关文本推荐来实现。比如,当用户在搜索引擎搜索“热度分析”一词时,相关的搜索词可能包括空间热度分析、关键词热度分析、音频热度分析、热词分析、关键词热度分析十法、网络游戏热度排行榜等。

相关文本推荐应用场景:新闻资讯推荐、博文帖子推荐、活动推荐、搜索内容推荐等。

相关文本推荐常用方法:Apriori、FP-Growth 等关联模型。

(6) 文档相似度判别

文档相似度是对不同文档间相似程度的判别,某些场景下应用非常广泛,尤其在文档、著作、报告、论文等科学研究和理论著作的原创性知识领域较为广泛,因此文档相似度识别是一个常用的场景。

文本相似度判别常用方法:余弦计算、隐形语义引标、基于语义相似度的文本相似度算法、基于拼音相似度的汉语模糊搜索算法、最长公共子序列、最小编辑距离算法等。

(7) 情感分析

情感分析是对情感倾向的分析,是用于分析特定对象对相关属性的观点、态度、情绪、立场以及其他主观感情的技术,通常分析的情感结果会属于正向、中性或负向。

情感分析应用场景:主要应用于竞争情报、舆情监测、客户正负向、话题监督、口碑分析等。

情感分析常用方法:除了非负矩阵分解、基于遗传算法的情感分析之外,使用最多的还是监督学习算法,例如朴素贝叶斯、K-近邻、最大熵和支持向量机等。

(8) 其他应用

除此以外,文本分析还广泛应用于简繁转换、自动注音、语音识别后处理、自动校对、机器翻译、基于句子的汉字键盘输入、用户兴趣模式识别等场景。

3. 图像分析

图像分析利用计算机对图像进行分析和处理,以达到识别不同模式和内容并得到对应分析结果的目的。图像分析中的图像识别是模式识别的一部分,也是人工智能的重要领域。图

像分析广义上包含图像识别和图像理解两个核心环节。

图像分析大致分为三个阶段：

第一个阶段是对语言文字的识别，这个阶段的典型技术是 OCR（Optical Character Recognition，光学字符识别），它将从借助于数码设备得到的图片中识别语言文字和数字等信息符号。该阶段已经相对成熟。

第二个阶段是对自然界所有存在的识别，包括动植物、客观物体、社会场景、人体特征、行为模式的识别。这是本节重点介绍的部分。该阶段处于发展初期，在某些领域已经有一些应用成果但尚未广泛推广到其他行业和领域。

第三个阶段是对图像中蕴含的深层意义进行理解，并以自然语言的形式进行表达，即透过表层图像的意义挖掘深层含义。该阶段目前还处于研究阶段。

图像分析的主要环节包括图像采集、图像预处理、特征提取、图像识别、图像理解和结果应用等。

图像采集

图像采集是数字图像数据提取的主要方式。数字图像主要借助于数字摄像机、摄像头、扫描仪、数码相机等设备经过采样数字化得到的图像，也包括从动态图像中获取的帧。

图像预处理

图像预处理是图像识别的重要步骤，其目的是去除干扰、噪声，将原始图像转换为适于计算机进行进一步处理的形态，主要环节包括图像复原、图像增强、归一化处理、图像分割、二值化、灰度化、图像编码压缩等。

（1）图像复原

由于在获取图像时环境光线、天气的影响，被摄物体的角度、运动，传感器问题，或因拍摄角度不对，会导致图像模糊不清，为了提取比较清晰的图像需要对图像进行复原。

图像复原主要采用滤波方法，根据图像退化的先验知识建立退化模型，然后采用反退化处理方法恢复原始图。图像复原的另一种特殊技术是图像重建，该技术是从物体横剖面的一组投影数据建立图像。

（2）图像增强

图像增强的目的是提高图像质量，尽可能地使其中的特定图像区域或信息更加明显或易于识别。它可以扩大图像中不同物体之间的差异性，以此来满足不同图像分析的需要。通过图像增强可以强调或减少图像中的特定内容，也可能会使图像有一定的失真或质量下降。

图像增强的常用方法包括直方图均衡化、对比度线性展宽、动态范围非线性调整、伪彩色增强、直方图匹配等。

（3）归一化处理

从不同来源得到的图像可能具有不同的标准，此时需要将图像的大小进行归一化变换和处理，以得到相同形式、尺寸或标准的图像。归一化通过保持仿射不变性来消除图像的几何变换（位移变化、切向变化、缩放变化、旋转变换）对识别带来的影响。

常用的图像归一化方法包括 MSER、Harris-Affine、Hessian-Affine 等。

(4) 图像分割

图像分割是将图像分为若干个相互独立的图像区域的过程,各个区域内部的颜色、形状、纹理等特征具有连续性和一致性。图像分割用于将图像中的目标、前景、背景等内容进行分离。图像分割可以减少大量的原始信息,提高对数据处理的实时性,加速后续处理的速度,提高图像特征提取和匹配的精度。比如,对车牌的识别只需要提取车牌信息即可,而不需要车身和环境等信息。

图像分割的主要方法有阈值分割方法、边缘检测方法、区域分割方法、聚类图像分割方法、小波变换分割方法、基于神经网络和遗传算法的分割方法等。

(5) 二值化

图像的二值化就是将图像中的每个像素点的灰度值设置为 0 或 255,最终呈现的效果是图像只有黑白两种颜色。图像二值化大量降低了图像计算量,并且能在一定程度上去除噪声,仅保留目标图像。

二值化可以将图片中的前景(或特定图像内容)与后景(其他图像内容)分离出来。大于阈值的像素群被赋予一种颜色(黑色或白色),小于阈值的像素群被赋予另一种颜色(白色或黑色)。如图 8-3 中的三幅图分别是原图、阈值为 128 的二值化图、阈值为 100 的二值化图。



图 8-3 风景原图和二值化图

二值化的关键是确定阈值,方法包括:灰度平均值法、百分比阈值(P-Tile 法)、基于谷底最小值的阈值、基于双峰平均值的阈值、迭代最佳阈值、OTSU 法、Kittle、一维最大熵、力矩保持法、基于模糊集理论的阈值等。

(6) 灰度化

灰度图,又称灰阶图,是用灰度表示的图像。它只含亮度信息,其亮度由暗到明,亮度变化是连续的。和原彩色图像相比,灰度图不含色彩信息,故灰度化之后的图像所含信息量大大减少,图像处理计算量也相应大幅减少,方便后续计算。

常见的彩色颜色标准包括 RGB、CMYK、HSV、YUV 和 HLS 几种。除了 CMYK 是用于工业排版和印刷外,其他都是适用于电子输出的形式。由于 RGB 是电子输出做流行的标

准，因此这里重点介绍基于 RGB 转灰度的方法，其他的颜色标准都可以跟 RGB 之间进行转换求得灰度值。

RGB 是用红（Red）、绿（Green）、蓝（Blue）三种颜色通道的变化和叠加来形成颜色。这个标准几乎可以表示所有人类可以感知的色彩，也是显示器采用的主要颜色标准。对 RGB 格式的图像进行灰度化，常用方法如下：

Adobe Photoshop 灰度法, Adobe Photoshop 对各个像素的灰度值公式为：灰度值 = $(R^{2.2} * 0.2973 + G^{2.2} * 0.6274 + B^{2.2} * 0.0753)^{(1/2.2)}$ 。



提示 Photoshop 简称为 PS，是 Adobe Systems 开发和发行的图像处理软件，是目前图像编辑和处理领域的标准工作工具之一。

分量法灰度值，它是将三分量的亮度作为三个灰度图像的灰度值，并选择其中一种灰度图像。

最大值法灰度值，它是取像素中的 R 、 G 、 B 三个分量的亮度最大值作为灰度值，计算公式为：灰度值 = $\max(B+G+R)$ 。

平均值法灰度值，它是取出像素中的 R 、 G 、 B 三个分量的亮度，并直接求得每个像素平均值来得到灰度值，计算公式为：灰度值 = $(B+G+R)/3$ 。

加权平均值法，它是将 R 、 G 、 B 三个分量以不同的权重进行加权然后求平均值，常用的加权方法包括 OpenCV 开放库加权法、基于人体视觉识别的加权法、基于心理学的加权法等。其计算公式分别为：

❑ OpenCV 开放库加权法灰度值 = $0.072169B + 0.715160G + 0.212671R$ 。

❑ 基于人体视觉识别的加权法灰度值 = $0.11B + 0.59G + 0.3R$ 。

❑ 基于心理学的加权法灰度值 = $0.299R + 0.587G + 0.114B$ 及其他变体。

(7) 图像编码压缩

图像压缩是用较少的数据量来表示原有的像素矩阵的过程，这个过程也称为图像编码。图像压缩分为有损压缩和无损压缩。数字图像的显著特点是数据量庞大，需要占用相当大的存储空间，这为图像的存储、计算、传输等带来了不便。图像编码压缩技术可以减少图像的冗余数据量和存储器容量，并提高图像传输速度、缩短数据计算和处理时间。目前，主流的图像编码格式包括 JPG、PNG、BMP、GIF、PCX、TIFF、SVG、PSD、EXIF、TGA、FPX、CDR 等。图像压缩的方法包括熵编码法、行程长度编码、色度抽样法、变换编码法、聚类等。

(8) 其他

对于图像的处理还涉及非常多的其他技术和细节，例如反色（反相），滤波，颜色分离，色相、饱和度、对比度、亮度等调整，颜色标准模式转换，以及各种图像滤镜和效果处理等。

特征提取

在模式识别中，需要进行特征的抽取和选择，特征提取负责把能够充分表示该图像的特

征用数值的形式表达出来。

一般图像特征可以分为四类：直观性特征、灰度统计特征、变换系数特征与代数特征。

□ 直观性特征主要指几何特征，几何特征比较稳定，受人脸的姿态变化与光照条件等因素的影响小，但不易抽取，而且测量精度不高，与图像处理技术密切相关。

□ 灰度统计特征包括灰度均值、方差、熵、能量等，这些是灰度直方图中的特征量。

□ 变换系数特征指先对图像进行 Fourier 变换、小波变换等，得到的系数作为特征进行识别。

□ 代数特征是统计学习方法抽取的特征。代数特征具有较高的识别精度，代数特征抽取方法又可以分为两类：一种是线性投影特征抽取方法，例如主成分分析、Fisher 线性鉴别分析 (FLD)；另一种是非线性特征抽取方法，例如 SVM、KPCA、KFA 等。

图像识别

图像识别的常用方法包括统计法、句法识别方法、神经网络法、模糊模式识别法。

(1) 统计法

统计法也称为决策理论法，它经过对大量图像的统计分析，找出其中的规律并提取反映图像本质特点的特征来进行图像识别。它以数学上的决策理论为基础，建立统计学识别模型，因而是一种分类误差最小的方法。

常用的图像统计模型有贝叶斯模型、马尔可夫随机场模型、决策函数法、K-近邻分类法、支持向量机、特征分析法、模板匹配法、线性和非线性判别函数等。统计法理论上可以解决最优分类器的设计问题，但却面临更为困难的概率密度估计问题的限制；并且由于其忽略了被识别图像的空间结构关系，当图像非常复杂、类别数很多时，将导致特征数量的激增，最终的分类结果很难保证。

(2) 句法识别法

句法识别以形式语言作为理论基础。从识别过程看，它像统计模式识别中的监督式分类一样，从一组训练样本触发，推导出合适的描述语言。该方法是对统计识别方法的补充，在用统计法对图像进行识别时，图像的特征是用数值特征描述的，而句法识别方法则是用符号来描述图像特征的。它模仿了语言学中句法的层次结构，采用分层描述的方法，把复杂图像分解为单层或多层的相对简单的子图像，主要突出被识别对象的空间结构关系信息。

模式识别源于统计方法，而句法识别方法则扩大了模式识别的能力，使其不仅能用于对图像的分类，而且可以用于对景物的分析与物体结构的识别。但当存在较大的噪声干扰时，句法识别方法容易产生误判，因此难以满足分类识别精度和可靠度的要求。

(3) 神经网络方法

神经网络法是指用神经网络算法对图像进行识别的方法。神经网络系统是由大量的、简单的神经元，通过广泛地按照某种方式相互连接而形成的复杂网络系统。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的。句法识别方法侧重于模拟人的逻辑思维，而神经网络侧重于模拟和实现人的认知过程中的感知觉

过程、形象思维、分布式记忆和自学习自组织过程，与符号处理是一种互补的关系。

神经网络特别适合处理需要同时考虑许多因素和条件的问题以及信息不确定性（模糊或不精确）问题。在实际应用中，由于神经网络法存在收敛速度慢、训练量大、训练时间长，且存在局部最小、过学习与欠学习问题；另外，如何选择网络节点数、初始权值和学习步长等问题也使其难以适用于经常出现新模式的场合，因而其实用性有待进一步提高。

（4）模糊模式识别法

在现实世界中，模糊性和随机性是两大不确定性。尽管两者有本质的区别，但是二者之间却可以相互交叉。同一研究对象往往不仅含有模糊性而且含有随机性。比如，“好山”“好水”“好人”都是模糊概念，而“可能性很大”“可能性很小”都是指事件发生的“概率”。为了解决模糊模式识别的问题，形成了一门专门的研究领域——模糊模式识别。

比较成熟的理论和方法有最大隶属原则、基于模糊等价关系的模式分类、基于模糊相似关系的模式和基于择近原则的识别分类和模糊聚类等。应用模糊方法进行图像识别的关键是确定某一类别的隶属函数，而各类的统计指标则要由样本像元的灰度值和样本像元的隶属函数的值即隶属度共同决定。隶属度表示对象隶属某一类的程度。

图像理解

传统的图像理解大多借助于浅层的视觉特征，例如颜色、纹理、形状、轮廓等形成所谓的内容理解，这在实质上是计算机对图像内容的理解而非人类的理解。然而，一幅图像中包含的各种信息和潜在语义远远比视觉特征所能表达的要丰富得多。

图像语义是指用户能够从图像中所得到的信息，既包括对图像中存在的对象及对象之间空间关系的理解，也包括对隐含在图像背后更为丰富的概念和内容的感受。图像语义的提取方法包括四种：基于处理范围的方法、基于机器学习的方法、基于人机交互的方法、基于外部信息的方法。

（1）基于处理范围的方法

按照特征提取范围的大小分为基于区域的提取方法与基于全局的提取方法。基于区域的提取方法是在图像分割和对象识别的前提下进行，利用对象模板、场景分类器等，通过识别对象及对象之间的拓扑关系挖掘语义，生成对应的场景语义信息。

比如，先以颜色、形状等特征对分割后的图像区域进行聚类，形成少量 BLOB；然后通过 CMRM 模型计算出 BLOB 与某些关键词共同出现的联合概率。

（2）基于机器学习的方法

机器学习中有两类不同的学习方法，即概率学习（研究在样本数趋向无穷大时的状况）和统计学习（研究在样本数足够多时的状况），因此基于机器学习的语义特征提取方法也可以分成基于概率的方法和基于统计的方法。常用的机器学习算法包括马尔可夫模型、SVM、贝叶斯网络、神经网络等。

比如，基于已经有语义标签的图片数据，通过 SVM 模型对其进行模型训练，通过不断用置信度因子动态调整分类器以改进类预测的准确性，并积累新的语义信息。

（3）基于人机交互的方法

基于人机交互的语义提取方法包括图像预处理和反馈学习两个方面。早期的对图像库中的图像进行人工标注就是一种简单的图像预处理方式。反馈学习是在提取语义的过程中加入人工干预，通过用户与系统之间的反复交互改进图像语义的提取方法，建立和调整与图像内容相关联的高层概念，构建真实的语义映射网络。

比如，通过半自动的方式对图像语义进行标注。当用户针对关键字或示例进行查找时，可对系统给出的图像进行相关性判定，同时系统根据用户的反馈产生或修改图像语义标志。

（4）基于外部信息的方法

基于外部信息的语义提取方法是指从图像外部附加信息中获取与图像内容相关的语义信息，然后进行语义分析与理解，包括如下几项。

- 图像本身的属性，例如文件名、标题、链接地址；
- 图像周围文本中反映图像内容主题、主体、背景等的关键词句；
- 一些其他的元数据信息，例如图片像素、拍照对比度、快门速度等参数。

比如，在处理网页的图片语义提取时，可以将图片周围的相关描述文字进行文本解析、语义提取、自然语义理解等，然后基于这些信息，再结合其他的语义提取方法来获得综合的图片语义。

现有的图片语义理解集中在二维图像，更多维的图像集合（例如视频图像、三维立体图像等）则包含更丰富的语义关联知识，可充分提取有价值的信息实现语义化的图像理解，拓展图像理解的研究领域，目前已经成为图像语义分析的研究热点。

结果应用

基于图像识别和理解的主要应用场景如下：

- 识别特定的图像信息，例如文字识别、验证码识别、人脸识别、虹膜识别、车牌识别、指纹识别、卫星图云识别、遥感图片识别等；
- 基于图像信息进行关联信息检索，可以应用到图片搜索、相似图片推荐；
- 基于潜在的语义可以建立图片信息摘要和关键字，并基于语义理解结果（而非文件名或文件类型）建立图片检索引擎和知识库，也可以做自动图像信息标注；
- 基于图像理解的结果还可以对特定属性、行为进行判断，提取潜在可能发生的行为和事实，进行针对性的预判与提前应对等。

4. 视频分析

视频分析（Intelligent Video System, IVS 或 Content analyse, CA）是计算机图像视觉技术的一个分支。它是使用计算机图像视觉分析技术，通过将场景中的背景和目标进行分离，追踪并分析在摄像机场景内出现的目标。

视频分析与移动侦测的本质区别是前者可以准确识别出视频中真正活动的目标，而后者只能判断出画面变化的内容，无法区分目标和背景干扰。当前的视频分析大多都是基于监控和安防需要而推动的，因此其技术和场景多应用于此。

视频分析从工作内容上可分为视频源、目标检测、图像处理、图像识别、目标跟踪、视频分析和结果应用。

视频源

视频源通常都产生于摄像机或摄影机，来源包括地铁、公交车、公路、停车场、商场、住宅、监狱等监控设备。

目标检测

运动物体目标的监测，根据背景是否固定可分为针对静态背景的检测方法和针对动态背景的检测方法。

(1) 针对静态背景的检测方法

针对静态背景检测就是摄像机在整个监视过程中不发生移动，只有被监视目标在摄像机监控范围内运动，这个过程只有目标相对于摄像机的运动。常用的方法包括光流法、帧间差分法、背景减除法等。根据实际应用需求的不同，不同的检测算法都是在可靠性、实时性和准确性之间折中得到的。其中，基于背景减除法是目前主要的目标检测方法。

1) 光流法。视觉心理学认为人与被观察物体发生相对运动时，被观察物体表面会带有光学特征，这些特征给人们提供了运动和结构的信息。当相机与场景目标间有相对运动时所观察到的亮度模式运动称之为光流（Optical Flow），或者说物体带光学特征部位的移动投影到视网膜平面（即图像平面）上就形成了光流。光流场的计算一般分为四类：基于梯度的方法（Horn-Schunck 和 Lucas-Kanade）、基于匹配的方法、基于能量的方法和基于相位的方法。

光流法不需要预先知道场景的任何信息，就能够检测到运动对象，可处理背景运动的情况，但噪声、多光源、阴影和遮挡等因素会对光流场分布的计算结果造成严重影响；而且光流法计算复杂，很难实现实时处理。

2) 帧间差分法。帧间差分法帧是一种通过对视频图像序列中相邻两帧作差分运算来获得运动目标轮廓的方法，它可以很好地适用于存在多个运动目标和摄像机移动的情况。

这种方法的实质就是利用相邻帧图像相减来提取前景目标移动的信息。它算法简单且程序复杂度低，对环境变化不敏感，具有较强的适应性和鲁棒性特征；但它不能提取完整对象区域，只能提取边界，且强依赖于关联序列帧。


3) 背景减除法。背景减除法（也称背景差分法）是一种有效的运动对象检测算法，基本思想是利用背景的参数模型来近似计算背景图像的像素值，将当前帧与背景图像进行差分比较实现对运动区域的检测，其中区别较大的像素区域被认为是运动区域，而区别较小的像素区域被认为是背景区域。

背景建模是背景消除法关键的一步。背景建模就是要自动学习和监视视频中场景的背景情况，并且能够随着时间的推移，适应各种动态的背景。比如，背景出现雨、雪、雨、电、云、雾、霾等天气现象，水纹、波浪、摇摆的树枝、现场光线变化、飞鸟掠过、异物坠落等

自然情况变化, 摄像机抖动等异常情况, 系统将启动各种背景学习和适应功能, 将这些动态背景或新增背景过滤掉, 并更新背景。

背景建模的常用方法包括:

- ❑ 单高斯分布模型: 单高斯分布模型将图像中每一个像素点的灰度值看成是一个随机过程 X , 并假设该点的某一像素灰度值出现的概率服从高斯分布, 那么就可以对每个像素点进行数学建模, 得到其期望值和偏差。
- ❑ 基于统计的方法: 该方法采用将颜色分解为亮度和色度的模型, 可以基于平均值、中位数、标准差等进行统计。核心是通过对视频中特定 N 帧图像的统计计算, 得到其对应的灰度值。优点是简单且速度快, 克服了单高斯模型对光照强度敏感度带来的问题。它可以解决全局和局部照明变化带来的问题, 但对环境光照变化和背景的多模态性比较敏感, 因此只能适用于静态背景。
- ❑ 混合高斯模型: 混合高斯模型通过对多个高斯概率密度函数的加权平均来平滑地近似任意形状的密度分布函数, 它可以将背景图像的每一个像素点按多个高斯分布的叠加来建模, 每种高斯分布代表一种背景场景, 因此, 多个高斯模型混合使用就可以模拟出复杂场景中的多模态情形。
- ❑ 概率密度估计模型: 高斯背景模型对像素点值的概率密度分布做了假设, 而这个假设不一定成立。非参数化背景模型不对像素点值做任何假设, 而通过概率密度估计的方法建立像素的背景统计模型。其基本思想是: 为被建模场景中的像素点保存一段时间内的一系列颜色样本值, 并根据这些样本值来估计当前帧图像中每一个像素点的概率。
- ❑ 卡尔曼滤波器模型: 该算法把背景认为是一种稳态的系统, 把前景图像认为是一种噪声, 用基于 Kalman 滤波理论的时域递归低通滤波来预测变化缓慢的背景图像, 这样既可以不断地用前景图像更新背景, 又可以维持背景的稳定性, 消除噪声的干扰。
- ❑ CodeBook 时间序列模型: CodeBook 时间序列模型是选择一帧到多帧使用更新算法建立 CodeBook 背景模型, 然后监测前景, 通过一定时间间隔使用算法更新 CodeBook 模型, 并对 CodeBook 进行时间滤波。这种模型能很好地处理时间起伏, 缺点是需要消耗大量的内存。
- ❑ ViBe 检测方法: ViBe 方法的具体思想是为每个像素点存储一个样本集, 样本集中的采样值就是该像素点过去的像素值和其邻居点的像素值, 然后将每一个新的像素值和样本集进行比较来判断是否属于背景点。其优点是减少了背景模型建立的过程, 还可以处理背景突然变化的情况, 缺点是容易引入拖影区域。
- ❑ W4 模型: Haritaoglu 等提出的 W4 模型将背景模型中的每个像素点用 3 个值来描述: 最大灰度值、最小灰度值和最大邻间差分。其中, 最大邻间差分值为相邻两帧的像素灰度差异最大值, 这 3 个参数需要靠事先存入的 L 帧背景图像来估计。

 **注意** 由于场景的复杂性、不可预知性，以及各种环境干扰和噪声的存在，使得背景的建模和模拟变得比较困难。

背景减除法检测运动目标速度快，检测准确，易于实现；但在实际应用中，静止背景是不易直接获得的，同时，由于背景图像的动态变化，需要通过视频序列的帧间信息来估计和恢复背景，即背景重建，所以要选择性地更新背景。

背景减除法、帧间差分法、光流法的简单对比，如表 8-5 所示。

表 8-5 三种检测方法对比

	背景减除法	帧间差分法	光流法
目标信息	运动对象位置、大小、形状等信息	运动对象边界信息	运动对象位置、大小、形状等信息
适用范围	固定摄像机（背景相对固定）	固定摄像机（背景相对固定）	固定或移动摄像机
方法核心	背景建模	相邻帧、目标运动速度	计算光流场
复杂性	由背景建模复杂程度决定	小	大
鲁棒性	较好	好	差
实时性	由模型复杂度决定	好	差

（2）针对动态背景的检测方法

针对动态背景检测是摄像机在整个监视过程中发生了移动（例如平移、旋转或多自由度运动），被监视目标在摄像机“视觉”范围内捕获到移动，这个过程就产生了目标与摄像机之间复杂的相对运动。

由于目标与摄像机之间存在相对运动，动态背景下运动目标检测要比静态背景下的目标检测复杂。通常情况下，摄像机的运动形式可以分为两种：

- ❑ 摄像机的支架固定，但摄像机可以偏转、俯仰以及缩放；
- ❑ 将摄像机装在某个移动的载体上。

由于以上两种情况下的背景及前景图像都在做全局运动，要准确检测运动目标的首要任务是进行图像的全局运动估计与补偿。

考虑到图像帧上各点的全局运动矢量不尽相同（摄像机做平移运动除外），但它们均是在同一摄像机模型下的运动，因而应遵循相同的运动模型，可以用同一模型参数来表示。

全局运动的估计问题可以归结为全局运动模型参数的估计问题，通常使用块匹配法或光流估计法来进行运动参数的估计。

基于块的运动估算和补偿是较为通用的算法。可以将图像分割成不同的图像块，假定同一图像块上的运动矢量是相同的，通过像素域搜索得到最佳的运动矢量估算。块匹配法主要有如下三个关键技术：

- ❑ 匹配法则，例如最大相关、最小误差等。

□ 搜索方法, 例如三步搜索法、交叉搜索法等。

□ 块大小的确定, 例如分级、自适应等。

图像处理

图像处理可以分为预处理和后处理。由于视频处理的最小粒度仍然以帧为单位, 而一帧就是一个相对独立的图像。因此, 基于图像分析的图像预处理内容, 同样适用于本节内容。除了在图像预处理中已经介绍过的图像去燥、二值化、灰度化等内容外, 在此介绍另外三个视频图像处理的重要工作: 阴影减除、形态学处理和连通性分析。

(1) 阴影减除

通过前面的背景建模和图像检测, 在检测出的图像中可能会存在阴影的干扰, 当阴影面积较大时还会覆盖邻近的目标 (例如交通摄像中相邻的两辆车), 致使算法误将多个运动目标检测为一个, 对后期识别造成困难, 使系统的整体性能下降, 所以必须消除阴影。

常用的阴影检测算法包括: 基于亮度的阴影检测算法、基于 HSV 空间的阴影检测算法、高斯阴影检测方法、基于归一化互相关函数的阴影检测算法。

□ 基于亮度的阴影检测算法。由于阴影覆盖区域的亮度值比背景相应区域的亮度值低, 因此当前帧和背景帧亮度差分为正数的像素不可能是阴影。当前景像素亮度值比背景像素点的亮度值低于一定值 T 时, 则认为该像素点可能为阴影。

□ 基于 HSV 空间的阴影检测算法。HSV 接近人的色觉, 且能精确反映一些灰度和色彩等信息。基于 HSV 空间阴影检测法是将阴影的像素值与该点的背景像素值进行比较, 若包含的相应色彩值和灰度值在一定的阈值下, 则该点是阴影。

□ 高斯阴影检测方法。该方法是基于 HSV 通道获得的颜色值进行统计, 在此基础上在三个通道上使用阴影样本训练模型参数建立高斯阴影模型。在阴影模型的基础上, 提出一种基于统计模型的前景阴影检测算法, 在各个通道上计算当前像素与阴影模型的匹配程度, 对各通道的匹配值进行加权, 与预定阈值进行比较从而判定是否为阴影。

□ 基于归一化互相关函数的阴影检测算法。像素点 (X, Y) 在未被阴影覆盖和被阴影覆盖时的亮度值近似呈线性关系, 因此可利用相关系数的性质进行阴影检测。在信号处理中, 归一化互相关函数 (Normalized Cross Correlation, NCC) 常被用来衡量两个信号间的相似性, 信号越相似, 则其 NCC 值越接近于 1。

(2) 形态学处理

在目标提取之后, 检测到的目标区域可能存在孔洞、边缘区域可能不平滑或出现横向和纵向的断层以及出现部分噪声图像, 这些都需要进行形态学的处理。形态学运算的主要方法包括开运算、闭运算、膨胀、腐蚀等。

形态学滤波常用方法有以下几种: 均值滤波、中值滤波、高斯滤波、带通 (包括低通、高通、带通和带阻) 滤波、数学形态学滤波等。不同滤波对噪声有不同的滤除效果: 高斯滤波对高斯噪声的处理效果最好; 中值滤波可有效地滤除脉冲型噪声, 而且对图像的边缘有较

好的保护；带通滤波对图像有平滑和锐化作用；形态学滤波对随机噪声有很好的滤除效果，而且可以分割或连接相邻区域。

（3）连通性分析

在经过形态学运算之后，图像中一些小的空洞被填充，小的间隙被连接，但是一些比较大的空洞或多目标的粘连性问题无法准确处理，这时候需要用到连通性分析来填补这些空洞或者断开多目标之间的粘结，最后得到较为干净的前景区域。

对于粘连性区域，除了一些真正的目标区域或背景区域外，它们不是真正的目标区域或背景区域，而是噪声点的连通集合。可以通过对产生的所有区域计算其像素值的大小（区域面积），如果超过某一阈值才判断其为目标区域，借以消除这些噪声区域的影响。



注意 数学形态学是一种应用于图像处理和模式识别领域的新理论和新方法。它主要以积分几何、几何代数及拓扑论为理论基础，将对象模型化，对集合进行研究。它认为集合结构等信息存在于对象之间的关系可通过结构元素联系，用具有一定形态的结构元素去度量和提取图像中的对应形状，以达到图像分析和识别的目的。数学形态学的基本思想是用具有一定形态的结构元素去度量和提取图像中的对应形状以达到对图像分析和识别的目的。数学形态学的数学基础和所用语言是集合论。数学形态学的应用可以简化图像数据，保持它们基本的形状特性，并除去不相干的结构。它在图像处理中可以应用到以下几个方面：图像预处理（去噪声、简化形状）、图像增强（抽取骨骼、细化、粗化、凸包、物体标记）、图像分割、边缘检测、特征抽取（面积、周长、投影、Euler-Poincare 特征）等。

图像识别

图像识别主要是识别图像中特定的属性和特征，包括人脸识别、车牌识别、物体识别（例如刀具、枪支、危险品）等。

目标跟踪

与目标检测相比，目标跟踪属于更高级的计算机视觉问题，它为下一步的行为理解提供数据。运动目标的跟踪是在一段序列图像中的每幅图像中实时地找到所感兴趣的运动目标（包括位置、速度及加速度等运动参数）。简单来说，就是在序列图像中为目标定位。在运动目标跟踪问题的研究上，总体来说有两种思路：

- 第一种是不依赖于先验知识，直接从图像序列中检测到运动目标，并进行目标识别，最终跟踪感兴趣的运动目标（该部分内容请见 8.4.3 节“目标检测”部分）；
- 第二种是依赖于目标的先验知识，首先为运动目标建模，然后在图像序列中实时地找到相匹配的运动目标。

（1）目标跟踪的核心环节

在目标跟踪中，首先需要明确目标的特征，即用哪些图像特征或变换数据来表示目标；

其次要通过一定的相似性度量和搜索算法,来识别不同序列帧中的目标形态和关联属性,以得到运动目标在不同帧的具体运动数据。因此,涉及目标特征表达、相似性度量算法和图像搜索算法三个关键技术点。

1) 目标特征表达。目标特征主要包括视觉特征(图像边缘、轮廓、形状、纹理、区域)、统计特征(直方图、各种图像矩特征)、变换系数特征(傅里叶描绘子、自回归模型)、代数特征(图像矩阵的奇异值分解)等。除了使用单一特征外,也可通过融合多个特征来提高跟踪的可靠性。

2) 相似性度量算法。对运动目标进行特征提取之后,需要采用一定的相似性度量算法与帧图像进行匹配,从而实现目标跟踪。常见的相似性度量方法有欧氏距离、曼哈顿距离、切比雪夫距离、汉明距离、加权距离、巴特查理亚系数、杰卡德相似系数等,其中应用最多和最简单的是欧氏距离。

3) 图像搜索算法。目标跟踪过程中,直接对场景中的所有内容进行匹配计算,寻找最佳匹配位置,需要处理大量的冗余信息,这会导致运算量比较大。采用一定的搜索算法对未来时刻目标的位置状态进行估计假设,缩小目标搜索范围便具有了非常重要的意义。

其中一类比较常用的方法是预测运动体下一帧可能出现的位置,在其相关区域内寻找最优。常见的预测算法有 Kalman 滤波、扩展的 Kalman 滤波及粒子滤波方法等。

□ Kalman 滤波器及其扩展算法是一个对动态系统的状态序列进行线性最小方差估计的算法。它通过状态方程和观测方程来描述一个动态系统,基于系统以前的状态序列对下一个状态作最优估计,预测时具有无偏、稳定和最优的特点,且具有计算量小、可实时计算的特点,可以准确地预测目标的位置和速度,但其只适合于线性且呈高斯分布的系统。

□ 粒子滤波器特别适用于非线性、非高斯的系统。粒子滤波算法是一种基于蒙特卡洛和贝叶斯估计理论的最优算法,它以递归的方式对测量数据进行序贯处理,因而无需对以前的测量数据进行存储和再处理,节省了大量的存储空间。在跟踪多形式的目标以及在非线性运动和测量模型中,粒子滤波器具有极好的鲁棒性。

另一类减小搜索范围的算法是优化搜索方向。均值漂移算法(Meanshift 算法)、连续自适应均值漂移算法(Camshift 算法)都是利用无参估计的方法优化目标模板和候选目标距离的迭代收敛过程,以达到缩小搜索范围的目的。

□ Meanshift 算法是利用梯度优化方法实现快速目标定位,能够对非刚性目标实时跟踪,适合于非线性运动目标的跟踪,对目标的变形、旋转等运动有较好的适用性。但是 Meanshift 算法在目标跟踪过程中没有利用目标在空间中的运动方向和运动速度信息,当周围环境存在干扰时(例如光线、遮挡),容易丢失目标。

□ Camshift 算法是在 Meanshift 算法的基础上,进行了一定的扩展,结合目标色彩信息形成的一种改进的均值漂移算法。由于目标图像的直方图记录的是颜色出现的概率,

这种方法不受目标形状变化的影响,可以有效地解决目标变形和部分遮挡的问题,且运算效率较高,但该算法在开始前需要由人工指定跟踪目标。

(2) 目标跟踪的算法分类

依据运动目标的表达和相似性度量,运动目标跟踪算法可以分为四类:基于主动轮廓的跟踪、基于特征的跟踪、基于区域的跟踪和基于模型的跟踪。跟踪算法的精度和鲁棒性在很大程度上取决于对运动目标的表达和相似性度量的定义,跟踪算法的实时性取决于匹配搜索策略和滤波预测算法。

1) 基于主动轮廓的跟踪。Kass 等提出的主动轮廓模型,即 Snake 模型,是在图像域内定义的可变形曲线,通过对其能量函数的最小化,动态轮廓逐步调整自身形状与目标轮廓相一致,该可变形曲线又称为 Snake 曲线。Snake 技术可以处理任意形状物体的任意形变,首先将分割得到的物体边界作为跟踪的初始模板然后确定表征物体真实边界的目标函数,并通过降低目标函数值,使初始轮廓逐渐向物体的真实边界移动。

基于主动轮廓跟踪的优点是不但考虑来自图像的灰度信息,而且考虑整体轮廓的几何信息,增强了跟踪的可靠性。由于跟踪过程实际上是解的寻优过程,带来的计算量比较大;另外,由于 Snake 模型的盲目性,对于快速运动的物体或者形变较大的情况,跟踪效果不够理想。

2) 基于特征的跟踪。基于特征匹配的跟踪方法不考虑运动目标的整体特征,只通过目标图像的一些显著特征来进行跟踪。假定运动目标可以由唯一的特征集合表达,搜索到该相应的特征集合就认为跟踪到了运动目标。除了用单一的特征来实现跟踪外,还可以将多个特征信息融合在一起作为跟踪特征。

基于特征跟踪方法的优点:

- 由于使用的符号模型运动方式简单,运动具有平滑性,因此跟踪目标的算法就简单了;
- 这种方法已经假设特征符号运动是相互独立的运动,因此在运动分析时可以不区分运动物体是刚体还是非刚体,也无需关注其几何形状;
- 跟踪过程中符号特征容易捕捉,能够匹配到每一个特征符号;
- 对运动目标的尺度、形变和亮度等变化不敏感,即使目标的某一部分被遮挡,只要还有一部分特征可以被看到,就可以完成跟踪任务;
- 这种方法与 Kalman 滤波器联合使用,具有很好的跟踪效果。

基于特征跟踪算法的缺点:

- 伴随着复杂运动的简单运动,刚体运动目标的特征提取就会产生困难;
- 刚体的一些特征会因为遮挡而无法识别,因此,基于特征的跟踪算法必须解决目标跟踪过程中运动初始化的难点,但这些问题的解决又会使跟踪算法变得非常复杂;
- 在改变符号参数和 3-D 目标运动参数时,这些参数是非线性的,因此特征跟踪中恢复的 3-D 运动参数对噪声相当敏感;

□ 连续帧间的特征对应关系也较难确定,尤其是当每一帧图像的特征数目不一致、漏检、特征增加或减少等情况出现时。

基于特征的跟踪主要包括特征提取和特征匹配两个方面。

□ 特征提取。特征提取是指从景物的原始图像中提取图像的描绘特征,图像特征具有直观性和较好的分类能力,计算相对简单,具备图像平移、旋转、尺度变化等不变性。目标跟踪中常用的特征主要包括颜色、纹理、边缘、块特征、光流特征、周长、面积、质心、角点等。常用的特征提取方式是基于特征空间的方法,包括 PCA、ICA、SCA 和 NMF 等。

□ 特征匹配。特征提取的目的是进行帧间目标特征的匹配,并以最优匹配来跟踪目标。常见的基于特征匹配的跟踪算法有基于二值化目标图像匹配的跟踪、基于边缘特征匹配或角点特征匹配的跟踪、基于目标灰度特征匹配的跟踪、基于目标颜色特征匹配的跟踪等。

3) 基于区域的跟踪。基于区域的跟踪算法的基本思想是:首先获取包含目标的模板,该模板通过图像分割或预先人为提取来确定,然后在序列图像中运用相关法搜索目标,从而确定目标在序列图像中的坐标位置。

这种算法的优点在于由于提取了较完整的目标模板,当目标未被遮挡时,跟踪精度非常高、跟踪非常稳定。但其缺点首先是在图像上遍历搜索,计算量大、比较耗时,当搜索区域较大时情况尤其严重;其次,算法要求目标变形不大,且不能有太大遮挡,否则相关精度下降会造成目标的丢失。

4) 基于模型的跟踪。基于模型的跟踪是通过一定的先验知识对所跟踪目标建立模型,然后通过匹配跟踪目标进行模型的实时更新。对于刚体目标来说,其运动状态变换主要是平移、旋转等,可以利用该方法实现目标跟踪。但是实际应用中跟踪的不仅仅是刚体,还有一大部分是非刚体,目标确切的几何模型不容易得到。

这种方法不易受观测视角的影响,具有较强的鲁棒性,模型匹配跟踪精度高,适合于机动目标的各种运动变化,抗干扰能力强;但其计算分析复杂、运算速度慢,模型的更新较为复杂,实时性较差。

视频分析

(1) 视频分析的两种实现方式

从实现方式来看,视频分析技术主要有两种常用方式:第一种是基于智能视频处理器的前端解决方案;第二种是基于监控的后端智能视频分析解决方案。

1) 基于智能视频处理器的前端解决方案。所有的目标跟踪、行为判断、报警触发都是由前端智能分析设备完成,只将报警信息通过网络传输至监控中心。优点是视频分析设备被放置在 IP 摄像机之后,这样可以有效地节约视频流占用的带宽;缺点是价格昂贵、安装复杂,前端设备分散、易损率高,报警记录与视频监控设备分离。

2) 基于监控的后端智能视频分析解决方案。在该方案中,所有的前端摄像机仅仅具备

基本的视频采集功能，而所有的视频分析都汇总到后端由计算机统一处理。优点是无需红外传感器等前端检测设备，可有效地与现有监控系统、预警等系统融合应用，且针对不同需求规则可灵活修订配置，可扩展性强；缺点是只能控制若干关键的监控点，并且对计算机性能和网络带宽要求比较高。

（2）视频分析的内容

从广义上来说，视频分析技术主要包括模式理解、视频分析、识别改善三类。

1) **模式理解**。模式理解是对图像理解的扩展，它通过分析目标的运动模型，生成对目标的个体行为和交互行为的高级描述，并用自然语言来表达这种描述。其重点是在图像分析的基础上进一步研究图像中各目标的性质及其相互关系，并得出对图像内容含义的理解以及对原来客观场景的解释，进而指导和规划行为。

举例，使用隐马尔可夫模型（HMM）对交通异常行为进行识别。首先通过目标检测与跟踪，提取运动特征作为基本语义单元，并将其组合成简单行为观测序列，然后利用 Baum-Welch 算法进行异常行为的 HMM 建模，最后根据前向算法及阈值处理识别交通异常行为。通过该算法能够准确识别出交通场景中车辆逆行、压线、异常停驶及行人翻越护栏等异常行为，并且实时性较好。

除此以外，模式理解还可以应用到物品移动检测、可疑行为监测、犯罪线索搜索、灾害的检测等场景。

2) **视频分析**。视频分析主要应用在监控领域，核心是在监控图像中找出目标，并检测目标的运动特征属性。在视频分析的过程中，基于目标检测、图像处理、图像识别和分类等，通过将这些信息与预先设定的报警规则、事件规则或阈值进行对比，即增加逻辑判断过程，可以实现对如下方面的跟踪和分析：

- 出入检测：检测是否有目标进入或离开指定的检测区域。
- 出现消失检测：检测目标是否出现或消失在指定检测区域。
- 方向检测：检测目标的移动方向或轨迹。
- 速度检测：监测目标的实时移动速度。
- 属性检测：检测目标的属性是否发生变化，例如大小、颜色、完整度、密度、比例等。
- 计数功能：可以根据指定的区域或者划线，设定各类规则进行精确计数。
- 绊线：监测是否有目标在指定监控区域内跨越（单向或双向）定义的一条线段。
- 自动跟踪：控制 PTZ 球机自动追踪目标。
- 物品监测：监测区域内是否存在固定模式的物品，例如明火、烟雾、遗弃物等。

3) **视频改善**。该类分析主要针对一些不可视、模糊不清，或者是对振动的图像进行部分优化处理，以增加视频的可监控性能。视频改善类主要基于监控到的视频，通过相应的视频处理技术进行处理（详见 8.4.3 节“图像处理”部分），具体包括：

- 视频稳定：将输入的不稳定视频（手持、车载、高空抖动）稳定后输出。
- 遮挡检测：将检测摄像机方向是否被改变、是否被遮挡、焦距是否被改变；如果是则

针对性地调整监测角度、方向、聚焦、景深等。

❑ 视频处理: 红外夜视图像增强处理、车牌识别影像消模糊处理、光变与阴影抑制处理、潮汐与物体尺寸过滤处理, 以及去除视频中的雾霾等。

结果应用

视频分析的结果, 可以广泛应用到各个行业和领域。

(1) 自助银行监管

智能视频分析可以有效地提升无人值守自助银行的防护水平, 自助银行主要用到的监控方式有 ATM 机遗弃物和出钞口异常检测、键盘篡改检测、警戒线内多人检测、后台加钞口入侵检测、自助银行室内过夜检测、打架检测、操作人员面部异常检测。

(2) 智能交通管理

智能视频分析可以轻松监控摄像覆盖范围内所有车辆的行驶状态与运行轨迹, 快速分析出其是否违章, 通过对海量交通数据的比对、分析和研判, 实现指定车辆行驶路径、道路拥堵、流量规模等的交通监督管理。

(3) 安防刑侦协助

智能视频分析技术在安防领域的重要作用毋庸置疑的, 其可以从海量信息中迅速搜索到需要的信息。对于重大刑事案件而言, 可以协助警方尽快破案, 节省了公安干警的办案时间。同时, 可以实时汇总并综合分析各种公共安全数据和资料, 为执法人员快速准确应对突发事件提供科学依据。指挥人员也可以参照各种数据对不同来源的资料进行综合分析, 制作指挥图。

(4) 云服务模式

实现基于大数据的视频监控云服务, 让摄像机仅通过互联网就能连接云端的视频监控托管服务, 通过快速、智能地分析部署在云端的大数据, 为小型企业、零售商店、餐馆酒店等提供实时的监控视频和潜在的风险管理, 甚至能提供收费的基于视频内容的分析报告, 例如日常的客户数、平均队列长度等, 创造新的商业模式。

(5) 其他

除了在监控、安防、云服务等领域外, 视频分析结果还可以应用于无人机救援、智慧城市管理等, 基于摄像头的实时视频数据分析还是交通工具辅助驾驶的前提, 例如 ACC、自动泊车 (含半自动泊车), 甚至也是自动驾驶过程中模式识别的重要组成部分。

5. 其他计算

上面提到的语音、文本、图像、视频等非结构化计算主要还是围绕“人”产生的相关数据开展计算工作, 并且聚焦场景更多的是 To C (面向客户) 的部分, 落地应用也是围绕如何针对“人”提供针对性的应用和服务。但除此之外, 还有大量的非结构化数据是与“人”关联性较小, 并且应用到其他场景的, 例如环境保护 (如水资源保护)、天气预测 (如预测台风的产生地域)、工厂运维防护 (如设备生命周期和故障预测)、灾害防护 (如地震检测中的全波形反演)、比赛设备性能检测 (如 F1 赛车动态监控与性能提升)、电力特征计算和结果仿真模

拟（如潮流估计）等各个方面。

因此，除了文本、语音、图像、视频数据类型外，还可能包括针对地理空间信息、遥感卫星数据、机器日志、传感器数据、仪器测量数据等方面非结构化数据的计算，这些都是针对特定部门或行业的应用，其计算过程和流程也具有特殊性，在此不展开介绍。

8.4.4 深度挖掘学习

所谓深度挖掘学习，是通过进一步对现有数据规律的挖掘和未知数据规律的探索，达到“理解”数据的目的，实现对数据更深的推理、归纳、演进。其中，涉及的相关领域包括机器学习、深度学习、人工智能、数据挖掘、神经网络等。



注意 现有学习算法都需要在一定程度上依赖于人类的先验经验，对其进行模型选择、参数设置、模型调优等工作，以适应不同的应用需求和场景。因此，这类工作远远达不到跟人类智能类似或接近的“智能”程度。基于算法产生的所谓的智能系统只具备有限的学习能力，仅能满足企业日常自动化、智能化、批量化应用的部分需求。对相关“数据智能工作”的探索和研究，必将促使其进一步发展，待相关理论、技术和应用成熟后，数据价值的产生和实现都将源于数据本身，借助计算机的力量实现完全的自主和自我工作。

在深度挖掘学习方面，主要实现的计算场景包括：降维、回归、聚类、分类、关联、时间序列、异常检测、协同过滤、主题模型等。

1. 降维

在对海量数据或大数据进行数据挖掘时，通常会面临“维度灾难”，原因是数据集的维度可以不断增加直至无穷多，但计算机的处理能力和速度却是有限的；另外，数据集的大量维度之间可能存在共线性的关系，这会直接导致学习模型的健壮性不够，甚至很多时候算法结果会失效。因此，我们需要降低维度数量并降低维度间共线性的影响。

数据降维也被称为数据归约或数据约减，其目的是减少参与数据计算和建模维度的数量。数据降维的思路有两类：一类是基于特征选择的降维；一类是基于维度转换的降维。

基于特征选择的降维

基于特征的选择指的是根据一定规则和经验，直接选取原有维度的部分参与到后续的计算和建模过程，用选择的维度代替所有维度，这个过程不产生新的维度。这种方式的好处在于，所选择的维度保留了原有维度的业务含义，可以用于后续的知识模式解读和业务理解，从而保证了最终的可应用性。

基于特征选择的降维方法通常有四种：

- 经验法：通过操作者的以往经验、实际数据情况、业务理解程度等综合考虑选择。
- 测算法：通过不断测试多种维度选择参与计算，通过结果来反复验证和调整并最终找

到最佳特征方案。

- ❑ 基于统计分析的方法：通过相关性分析不同维度间的线性相关性，从相关性高的维度中人工去除或筛选；或者通过计算不同维度间的互信息量，找到具有较高互信息量的特征集，然后把其中的一个特征去除或留下。
- ❑ 机器学习算法：通过机器学习算法得到不同特征的特征值或权重，然后再根据权重来选择较大的特征。如图 8-4 所示是通过 CART 决策树模型得到不同变量的重要程度，然后可以根据实际权重值进行选择。

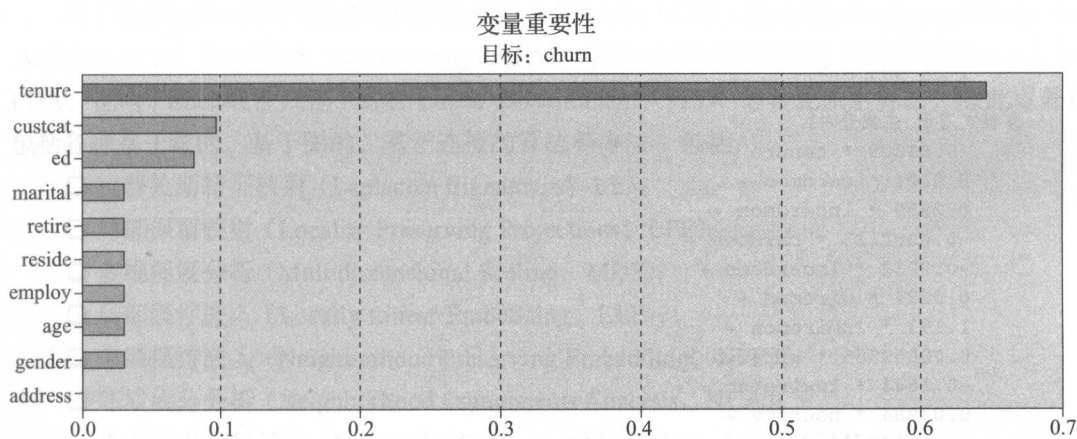


图 8-4 变量重要性得分

基于维度转换的降维

基于维度转换的降维是按照一定的数学变换方法，把给定的一组相关变量（维度）通过数学模型将高维空间的数据点映射到低维度空间中，通过映射后变量的特征来表示原有变量的总体特征。这种方式是一种产生新维度的过程，转换后的维度并非原有维度的本体，而是其综合转换或映射后的表达。

比如，假设原始数据集中有 10 个维度分别是 tenure、cardmon、lncardmon、cardten、lncardten、wiremon、lnwiremon、wireten、lnwireten、hourstv，现在用主成分分析进行降维，降维后的每个“成分”是：

方程式用于 主成分 -1

```
0.006831 * tenure +
0.007453 * cardmon +
0.1861 * lncardmon +
0.0001897 * cardten +
0.1338 * lncardten +
0.007552 * wiremon +
0.3688 * lnwiremon +
0.0001155 * wireten +
0.132 * lnwireten +
0.00006106 * hourstv +
```

+ -4.767

方程式用于 主成分 -2

-0.005607 * tenure +
0.03288 * cardmon +
0.759 * lncardmon +
0.0002219 * cardten +
0.05385 * lncardten +
-0.01013 * wiremon +
-0.4433 * lnwiremon +
-0.0001222 * wireten +
-0.1354 * lnwireten +
0.008099 * hourstv +
+ -0.272

方程式用于 主成分 -3

-0.01809 * tenure +
0.0124 * cardmon +
0.2859 * lncardmon +
-0.0002123 * cardten +
-0.2252 * lncardten +
0.0287 * wiremon +
1.193 * lnwiremon +
0.00002565 * wireten +
-0.1644 * lnwireten +
0.03984 * hourstv +
+ -4.076

方程式用于 主成分 -4

0.004858 * tenure +
-0.00553 * cardmon +
-0.1352 * lncardmon +
0.00005099 * cardten +
0.03662 * lncardten +
-0.006129 * wiremon +
-0.2493 * lnwiremon +
0.00001144 * wireten +
0.04165 * lnwireten +
0.198 * hourstv +
+ -2.994

方程式用于 主成分 -5

-0.01763 * tenure +
-0.005433 * cardmon +
0.5251 * lncardmon +
-0.001116 * cardten +
0.7426 * lncardten +
-0.004448 * wiremon +
0.7988 * lnwiremon +
-0.0005994 * wireten +
0.6897 * lnwireten +
0.00946 * hourstv +
+ -11.09

上述转换结果就是主成分分析后提取的5个能代表原始10个维度的新“维度”。通过上述结果可以发现主成分（也就是新的“维度”）是一个线性方程，而且是多元一次的方程；其含义是无法被人类直接理解的，这意味着人类无法将特征提取的方法直接应用到主成分分析之上（在类似于特征提取或分类器的算法中，可以直接了解原始参与计算的每个维度的权重的高低）。但对于不关注每个维度业务含义的应用来讲是没有关系的，因为机器学习或数据挖掘不需要知道每个因子的具体含义，只需要知道用哪些因子，如何计算便能得到最终结果。

基于维度转换的降维方法常用的算法如独立成分分析（ICA）、主成分分析（Principal Component Analysis, PCA）、因子分析（Factor Analysis, FA）、线性判别分析（Linear Discriminant Analysis, LDA，也叫 Fisher 线性判别 Fisher Linear Discriminant, FLD）等常见映射算法，除此以外还包括各种基于核的、基于图的、基于连接的算法等方法，包括：

- ☐ 拉普拉斯特征映射（Laplacian Eigenmaps, LE）;
- ☐ 局部保留映射（Locality Preserving Projections, LPP）;
- ☐ 多维标度分析（Multidimensional Scaling, MDS）;
- ☐ 局部线性嵌入（Locally Linear Embedding, LLE）;
- ☐ 邻域保持嵌入（Neighborhood Preserving Embedding, NPE）;
- ☐ 邻域成分分析（Neighborhood Components Analysis, NCA）;
- ☐ 最大坍塌度量学习（Maximally Collapsing Metric Learning, MCML）;
- ☐ 非负矩阵分解（Non-negative Matrix Factorization, NMF）;
- ☐ 等距映射（Isometric Mapping, ISOMAP）;
- ☐ Hessian 特征映射（Hessian Eigenmaps）;
- ☐ 基准点等距映射（Landmark ISOMAP）;
- ☐ 局部切空间排列（Local Tangent Space Alignment, LTSA）;
- ☐ 核主成分分析（Kernel PCA）;
- ☐ 概率主成分分析（Probabilistic PCA）;
- ☐ 核线性判别分析（Kernel LDA）;
- ☐ 局部线性坐标（Locality-constrained Linear Coding, LLC）;
- ☐ 自组织映射（Self-organizing map, SOM）;
- ☐ 偏最小二乘（Partial Least Squares, PLS）;
- ☐ 奇异值分解（Singular Value Decomposition, SVD）;
- ☐ 广义判别分析（Generalized Discriminant Analysis, GDA）;
- ☐ 最大方差展开（Maximum Variance Unfolding, MVU）;
- ☐ 松弛最大方差展开（Relaxed MVU）;
- ☐ 基准点最大方差展开（Landmark MVU）;
- ☐ 快速最大方差展开（Fast MVU）;

- ☐ 产生式拓扑映射 (Generative Topographic Mapping, GTM);
- ☐ 主曲线 (Principal Curves);
- ☐ Sammon 映射 (Sammon Mapping);
- ☐ 扩散映射 (Diffusion Maps);
- ☐ 半正定嵌入 (Semi-definite Embedding, SDE);
- ☐ 随机邻域嵌入 (Stochastic Neighbor Embedding, SNE);
- ☐ 对称随机邻域嵌入 (Symmetric Stochastic Neighbor Embedding, SymSNE);
- ☐ t-分布随机邻域嵌入 (t-Distributed Stochastic Neighbor Embedding, tSNE);
- ☐ 随机接近嵌入 (Stochastic Proximity Embedding, SPE);
- ☐ 线性局部切空间排列 (Linear Local Tangent Space Alignment, LLTSA);
- ☐ 共形特征映射 (Conformal Eigenmaps, CCA);
- ☐ 局部线性坐标 (Locally Linear Coordination, LLC);
- ☐ 流形坐标卡 (Charting a Manifold);
- ☐ 坐标化因子分析 (Coordinated Factor Analysis, CFA);
- ☐ 高斯过程隐变量模型 (Gaussian Process Latent Variable Model, GPLVM);
- ☐ 局部与全局保持嵌入 (Locality and Globality Preserving Embedding, LGPE);
- ☐ 大间隔最近邻域度量学习 (Large Margin Nearest Neighbor Metric Learning, LMNN)。

为了更形象地解释映射关系，现在使用一个序列图来表示整个映射过程。图 8-5 为原始数据集在经过顺时针旋转（映射变换）后的可视化特征变化过程。从图中可以看出，原始数据的分布呈现明显的曲线分布特征，而旋转后的数据样本则呈现直线分布特征。

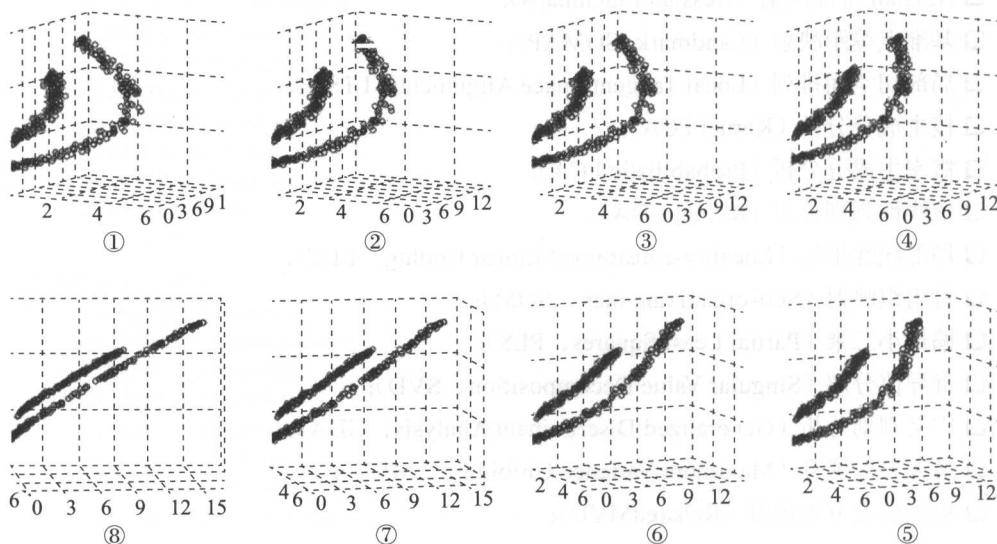


图 8-5 数据维度特征转换过程

2. 回归

回归是研究自变量 x 对因变量 y 影响的一种数据分析方法。最简单的回归模型是一元线性回归（只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示），可以表示为 $y = \beta_0 + \beta_1 x + \varepsilon$ ，其中 y 为因变量， x 为自变量， β_1 为影响系数， β_0 为截距， ε 为随机误差。

回归分析按照自变量的个数分为一元回归模型和多元回归模型；按照影响是否线性分为线性回归和非线性回归。如图 8-6 中左侧的费用与流量关系图是一个典型的线性回归图，显示了自变量 x （费用）与因变量 y （流量）之间的关系，二者的关系公式表示为 $y = 2.2072x + 1.6831$ ；右侧是一个典型的非线性回归图，显示了自变量 x （流量）与 y （CPC）之间的关系，二者的关系可用 $y = 8.8645x^{-1.28}$ 表示。

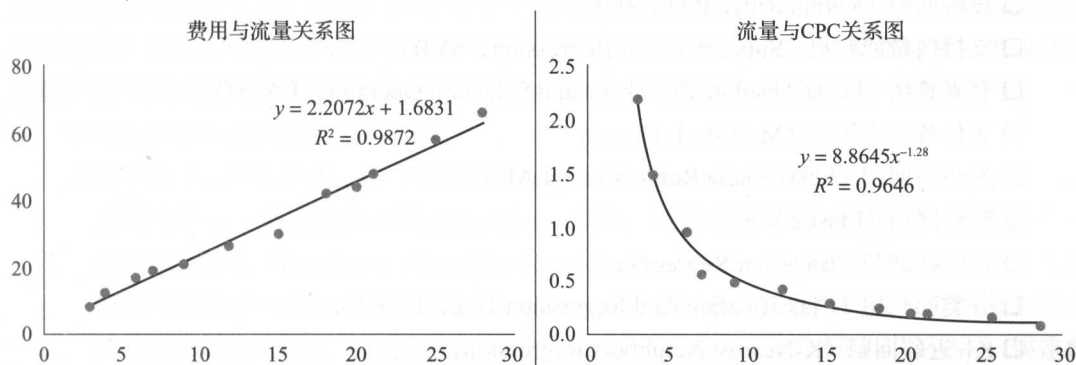


图 8-6 回归分析模型

在应用回归模型时，应注意解决自变量间的共线性问题（通过相关性分析得出）；另外，使用回归进行预测时，自变量 x 需要处于自变量拟合和计算的范围之内，避免预测时任意外推。

提示 细心的读者一定会发现图 8-6 中带有的“ R^2 ”参数， R^2 是解释系数，意思是拟合的模型中因变量能被解释的百分比，例如流量与 CPC 关系图中 $R^2=0.9646$ 的意思是拟合模型 $y = 8.8645x^{-1.28}$ 能解释 96.46% 的变化，还有 3.54% 的变化无法通过此模型解释。 R^2 越大代表解释模型越科学， R^2 取值区间在 0 ~ 1，即最大值为 1。

回归分析是广泛应用的统计学分析方法，可以用于分析其中一方对另一方的影响关系（通过自变量求因变量），也可以分析自变量对因变量的影响方向（正向影响还是负向影响）。回归分析的主要应用场景是进行预测和控制，例如计划制定、KPI 制定、目标制定等方面，也可以基于预测的数据与实际数据进行比对和分析，确定事件发展程度并给未来行动提供方向性。

回归分析的优点是数据模式和结果便于理解，例如线性回归用 $y = ax + b$ 的形式表达，在解释和理解自变量与因变量关系式时相对容易；在基于函数公式的业务应用中，可以直接通

过代入法进行求解，因此应用起来比较容易。回归分析的缺点是只能分析少量变量之间的相互关系，无法处理海量变量间的相互作用关系，尤其是变量共同因素对因变量的影响程度。

回归分析的算法除了常见的线性回归、逻辑回归（Logistics 曲线模型）、双曲线模型、二次曲线模型、对数模型、三角函数模型、指数模型、幂函数模型、修正指数增长模型外，还包括：

- ❑ 普通最小二乘法（Ordinary Least Square Regression, OLSR）;
- ❑ 偏最小二乘法（Partial Least Squares, PLS）;
- ❑ 逐步回归（Stepwise Regression）;
- ❑ 主成分回归（Principal Component Regression, PCR）;
- ❑ 岭回归（Ridge Regression）;
- ❑ 核岭回归（Kernel Ridge Regression）;
- ❑ 支持向量回归机（Support Vector Regression, SVR）;
- ❑ 套索算法（Least Absolute Shrinkage and Selection Operator, LASSO）;
- ❑ 多任务套索算法（Multi-task Lasso）;
- ❑ 最小角回归（Least Angle Regression, LARS）;
- ❑ 弹性网络（Elastic Net）;
- ❑ 贝叶斯回归（Bayesian Regression）;
- ❑ 分类回归树（Classification And Regression Tree, CART）;
- ❑ K-近邻回归（K-Nearest Neighbors Regression）;
- ❑ 随机梯度下降（Stochastic Gradient Descent, SGD）;
- ❑ 多元样条自适应回归（Multivariate Adaptive Regression Splines, MARS）;
- ❑ 局部多项式回归拟合（Locally Estimated Scatterplot Smoothing, LOESS）;
- ❑ 正交匹配跟踪（Orthogonal Matching Pursuit, OMP）;
- ❑ 保序回归（Isotonic Regression, IR）;
- ❑ 梯度推进回归（Gradient Boosting Regression Tree, GBRT）;
- ❑ GBDT（Gradient Boost Decision Tree, GBDT）;
- ❑ 人工神经网络回归（Artificial Neural Network Regression）;
- ❑ 泊松回归（Poisson Regression）。



很多算法通常是用来做分类学习的，例如支持向量机、决策树、K-近邻等，但经过改良之后，也可以应用于回归学习和分析。

3. 聚类

聚类是数据挖掘和计算中的基本任务，聚类是将大量数据集中具有“相似”特征的数据点划分为统一类别，并最终生成多个类的方法。聚类分析的基本思想是“物以类聚、人以群分”，因此大量的数据集中必然存在相似的数据点，基于这个假设就可以将数据区分出来，

并发现每个数据集(分类)的特征。图 8-7 为使用 DBSCAN 进行聚类的示意图。

聚类常用于数据探索或挖掘初期,在没有做数据整体分析之前进行的探索性分析,适用于样本量较大情况下的数据初步探索。比如,针对企业整体用户特征,在未得到相关知识或经验之前先根据数据本身特点进行用户分群,然后再针对不同群体做进一步分析。

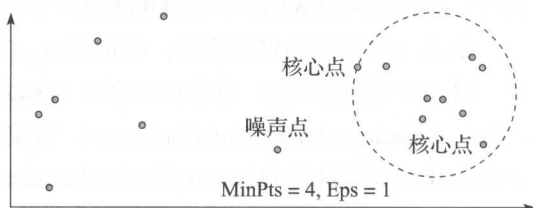


图 8-7 DBSCAN 聚类

聚类能解决的问题类型包括:目前的数据集可以分为几类,每个类别有多少样本量,不同类别中各个变量的强弱关系如何,不同类别的典型特征是什么等;同时,基于聚类可以对客户进行细分,在市场研究中的规模测试、机会发掘,以及进行图片压缩等应用。聚类无法提供明确的行动指向,聚类结果更多地是为后期挖掘和分析工作提供参考,无法回答“为什么”和“怎么办”的问题。

聚类的常用算法包括:

- ❑ 基于划分的聚类算法: K-均值(K-Means)、K-Modes、K-Medoids、K-Prototypes、最大期望算法(Expectation Maximization, EM)、均值偏移(Mean Shift)、基于随机选择的聚类算法(Clustering Algorithm based on Randomized Search, CLARANS)、大型应用中的聚类(Clustering LARge Applications, CLARA)、聚焦的大型应用中的聚类(Focused CLARAN)、围绕中心点的划分(Partitioning Around Medoid, PAM)、快速鲁棒 EM(Fast and Robust EM, FREM)、基于动态近邻选择模型的聚类算法(A Clustering Algorithm using Dynamic Nearest Neighbors Selection Model, DNNS)、流聚类(Streaming K-Means)、快速迭代聚类(Power Iteration Clustering, PIC)、模糊聚类(Fuzzy K-Means 或 Fuzzy C-Means, FCM)、可能性 C 均值(Possibility C-means, PCM)。
- ❑ 基于层次的聚类算法: CURE(Clustering Using Representatives, CURE)、BIRCH(Balanced Iterative Reducing and Clustering using Hierarchies, BIRCH)、ROCK(Robust Clustering using linKs, ROCK)、变色龙算法(A Hierarchical Clustering Algorithm using Dynamic Modeling, CHAMELEON)、基于相似度算法的聚类(Similarity-Based Agglomerative Clustering, SBAC)、BUBBLE、BUBBLE-FM、DIANA、AGNES、合成聚类(Agglomerative Hierarchical Clustering, AHC)。
- ❑ 基于密度的聚类算法: 基于密度的带有噪声的空间聚类(Density-based Spatial Clustering of Application with Noise, DBSCAN)、广义的 GDBSCAN(Generalized DBSCAN)、变密度 DBSCAN(Varied VDBSCAN)、基于网格和贡献的多密度 DBSCAN(Multi-density DBSCAN based on Grid and Contribution, GCMDDBSAN)、DBLASD、OPTICS(Ordering Points To Identify the Clustering Structure, OPTICS)、FDC、DENCLUE(DENsity based CLUstEring)。
- ❑ 基于网格的聚类算法: 基于网格统计信息的方法(Statistical Information Grid-based Method,

STING)、小波聚类 (Clustering with Wavelets, WaveCluster)、CLIQUE (Clustering In QUEst, CLIQUE)、OPTIGRID。

- ❑ 基于统计学的聚类算法：COBWeb、CLASSIT、AutoClass、P-AutoClass。
- ❑ 基于模型的聚类：混合高斯模型 (Gaussian Mixture Model, GMM)、SOM 神经网络 (Self Organizing Maps Neural Networks)、自组织特征映射 (Self-Organizing Feature Map, SOFM)、学习矢量化 (Learning Vector Quantization, LVQ)、竞争学习 (Competitive Learning)。
- ❑ 基于图论的聚类：仿射聚类 (Affinity Propagation clustering, AP)、谱聚类 (Spectral Clustering)。

4. 分类

分类算法通过对已知类别训练集的计算和分析，从中发现类别规则，以此预测新数据的类别的一类算法。分类算法是解决分类问题的方法，是数据挖掘、机器学习和模式识别中一个重要的研究领域。

与分类的概念类似的另一个概念是“聚类”，二者经常被混淆和混用，二者差异如下：

- ❑ 学习方式不同。聚类是一种非监督式学习算法，而分类是监督式学习算法。
- ❑ 对源数据集要求不同。聚类不要求源数据集有标签，但分类需要标签用来做学习。
- ❑ 应用场景不同。聚类一般应用于做数据探索性分析，而分类更多地用于预测性分析。
- ❑ 解读结果不同。聚类算法的结果是将不同的数据集按照各自的典型特征分成不同类别，不同人对聚类的结果解读可能不同；而分类的结果却是一个固定值 (类别)，不存在不同解读的情况。

分类的主要用途和场景是“预测”，基于已有的样本预测新的样本的所属类别。比如，信用评级、风险等级、欺诈预测等；它是模式识别的重要组成部分，例如机器翻译，人脸识别、医学诊断手写字符识别、指纹识别等图像识别领域，语音识别领域，视频识别领域等；另外，还可以用于知识抽取，通过模型找到潜在的规律，帮助业务得到可执行的规则。比如，图 8-8 所示是通过决策树产生的业务规则以及对应预测成功的概率。

id	段	得分	四分位数 (n)	频数	概率
	包括提示程序的所有段		13,504	1,952	14.45%
1	❑ income, number_transactions income > 55,267 和 number_transactions > 3	1	350	328	93.71%
2	❑ income, rfm_score income > 55,267 和 rfm_score > 10.53 和 rfm_score <= 12.33	1	218	193	88.53%
3	❑ income, average#balance#feed#index, call_cen income > 55,267 和 average#balance#feed#index > 124 和 average#balance#feed#index <= 349 和 call_center_contacts > 1 和 call_center_contacts <= 3	1	125	118	94.40%

图 8-8 决策树规则

其中“段”一列中是关于规则 (参与建模的维度结果) 的部分，里面的每个规则都可

以进行解读。以 ID1 为例,当选择特征符合收入 (income) > 55267 且订单次数 (number_transactions) > 3 的用户群体时,预计响应概率达到 93.71% (此时样本只有 350,意味着只能发送 350 个用户,预计其中有 328 个用户会响应)。

分类常用的算法包括:

- ❑ 贝叶斯类:朴素贝叶斯 (Naive Bayesian)、TAN (Tree Augmented Bayes Network, TAN)、贝叶斯网络分类器 (Bayesian Network Classifier, BNC)、多元自适应回归样条 (Multivariate Adaptive Regression Spline, MARS)、贝叶斯信念网络 (Bayesian Belief Network, BBN)、平均单依赖估计 (Averaged One-Dependence Estimators, AODE)。
- ❑ 决策树类:迭代二叉树 3 代 (ID3)、分类与回归树算法 (CART)、C4.5、C5.0、PUBLIC (Pruning and Building Integrated in Classification)、SLIQ、SPRINT、梯度推进机 (Gradient Boosting Machines, GBM)、单层决策树 (Decision Stump)、卡方自动互动检视 (Chi-squared Automatic Interaction Detection, CHAID)、随机森林 (Random Forest)、GBDT (Gradient Boost Decision Tree, GBDT)、PART 决策树、M5P、M5Rules、最小均方算法 (Least Mean Square, LMT)。
- ❑ 神经网络类:BP (Back Propagation) 神经网络、RBF (Radical Basis Function) 网络、Hopfield 网络、自组织特征映射神经网络 (SOFM 神经网络)、学习矢量化 (LVQ) 神经网络。
- ❑ 基于数据库的分类算法: MIND (Mining in Database)、GAC-RDB (Grouping and Counting-relational Database)。
- ❑ 其他常用算法: K-近邻 (K-Nearest Neighbor, KNN)、支持向量机 (Support Vector Machine, SVM)、逻辑回归 (Logistics Regression, LR)、遗传算法、CAEP (Classification by Aggregating Emerging Patterns)、隐马尔可夫模型 (HMM)、条件随机场 (CRF)、感知机 (Perceptron Learning Algorithm, PLA)、在线被动攻击算法 (Online Passive Aggressive Algorithms)、深度学习、判别分析 (Discriminant Analysis, DA)、OneR。

5. 关联

关联规则学习通过寻找最能够解释数据变量之间关系的规则,来找出大量多元数据集中有用的关联规则,它是从大量数据中发现多种数据之间关系的一种方法;另外,它还可以基于时间序列对多种数据间的关系进行挖掘。关联分析的典型案例是“啤酒和尿布”的捆绑销售,即买了尿布的用户还会一起买啤酒。

关联规则相对其他数据挖掘模型简单,易于业务理解和应用。关联模型的典型应用场景是购物篮分析,通过分析用户同时购买了哪些商品来分析用户的购物习惯。这种策略还会应用于捆绑销售、商品促销设计、页面促销设计、商品陈列设计、商品价格策略和基于购买的用户特征分析等。网站分析工具 Webtrekk 中的关联分析报表即应用了关联模型。

关联模型广泛应用于购物篮分析、网站页面浏览分析、广告流量来源分析、用户关键字搜索分析、网站内容或产品查看分析、生物特征提取、DNA 序列破译、自然灾害预测、科学

实验分析等，回答的问题类似于“发生了 A 之后，还会发生 B 还是 C？”或者“通常 A 和 B 还是和 C 一起发生”？

图 8-9 显示了购物篮中客户最经常购买的商品，水果蔬菜和鱼、啤酒和蔬菜罐头、白酒和糕饼的关联购买性较强。

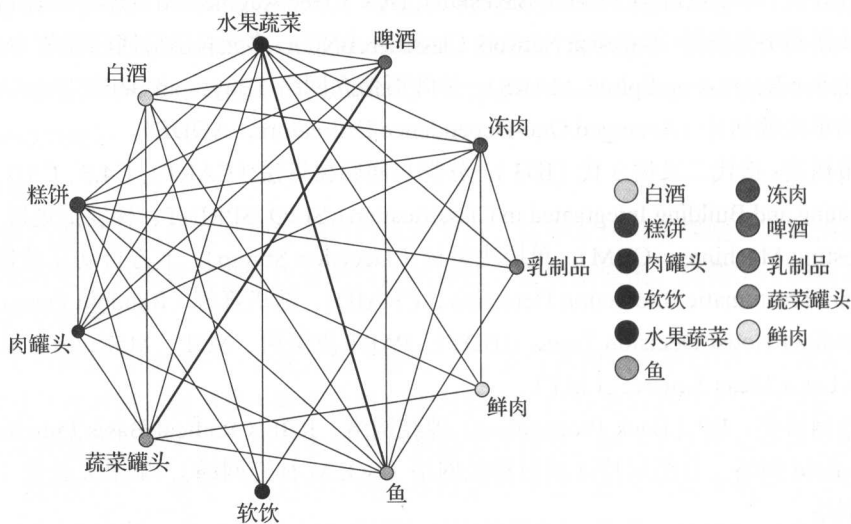


图 8-9 商品关联销售

常见的关联算法和序列关联算法包括：

- Apriori;
- FP-Growth;
- Eclat 和 Eclat+;
- PrefixSpan;
- SPADE 和 cSPADE;
- Tertius;
- AprioriAll;
- AprioriSome;
- GSP (Generalized Sequential Patterns, GSP)。

6. 时间序列

时间序列是用来研究数据随时间变化趋势而变化的一类算法，它是一种常用的回归预测方法。它的原理是事物的连续性，所谓连续性是指客观事物的发展具有合乎规律的连续性，事物发展是按照它本身固有的规律进行的。在一定条件下，只要规律赖以发生作用的条件不产生质的变化，则事物的基本发展趋势在未来就还会延续下去。

从本质上看，时间序列算法是利用统计技术与方法，从预测指标的时间序列中找出演变

模式,建立数学模型,对预测指标的未来发展趋势做出定量估计。如图8-10所示为使用移动平均算法对每日销售额进行拟合和预测。

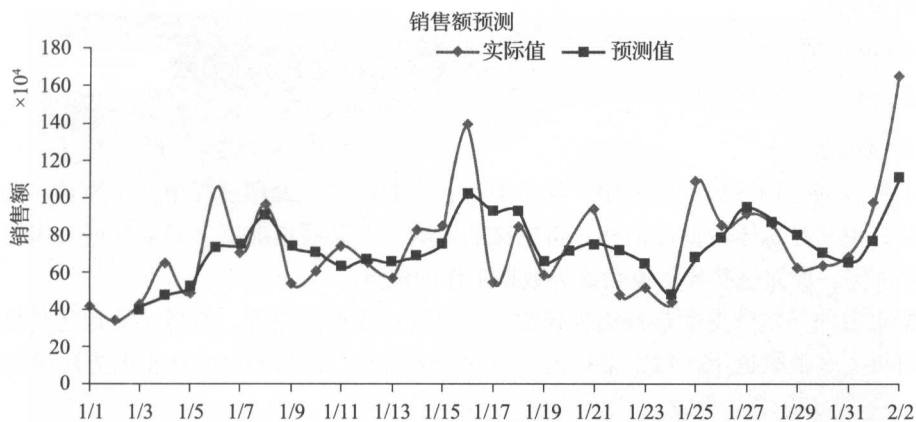


图 8-10 销售额预测

时间序列可以解决在只有时间(序列项)而没有其他可控变量下对未来数据的预测问题,常用于经济预测、股市预测、天气预测等偏宏观或没有可控自变量的场景下。

时间序列算法中通常需要包含四种变化要素:长期趋势(T)、季节波动(S)、循环波动(C)、不规则波动(I)。

- 长期趋势(T): 指数据随时间长期性的增长或下降的趋势;
- 季节变动(S): 指在一年内随季节周期的变动趋势,例如节假日、季节周期等;
- 循环变动(C): 指在若干年或长期时间内的变动趋势,与趋势的区别在于循环变动具有周期性规律,而非呈现单一方向的持续变化;
- 不规则变动(I), 指异常情况、突发性事件导致的异常变动。

注意 时间序列是用来做回归预测的,但与其他回归预测算法不同,时间序列缺乏可灵活选择的自变量,唯一的自变量是数据中可用的时间序列规则,例如分、小时、日、周、月、季度、年等。

时间序列的常用算法包括移动平均(Moving Average, MA)、指数平滑(Exponential Smoothing, ES)、差分自回归移动平均模型(Auto-regressive Integrated Moving Average Model, ARIMA)三大类,每个类别又可以细分和延伸多种算法,包括:

- 简单移动平均;
- 加权移动平均;
- 趋势移动平均;
- 简单指数平滑;

☐ Holts 线性趋势预测；

☐ Browns 线性趋势；

☐ 阻尼趋势；

☐ Winters 加法；

☐ Winters 乘法。

7. 异常检测

大多数数据挖掘或数据工作中，异常值都会在数据的预处理过程中被认为是“噪声”而剔除，以避免其对总体数据评估和分析挖掘的影响。但某些情况下，如果数据工作的目标就是围绕异常值，那么这些异常值会成为数据工作的焦点。

数据集中的异常数据通常被称为异常点、离群点或孤立点等，典型特征是这些数据的特征或规则与大多数数据不一致，呈现出“异常”的特点，而检测这些数据的方法被称为异常检测。



注意 “噪声”的出现有多种原因，例如业务操作的影响（典型案例如网站广告费用增加10倍，导致流量激增）、数据采集问题（典型案例如数据缺失、不全、溢出、格式匹配等问题）、数据同步问题（异构数据库同步过程中的丢失、连接错误等导致的数据异常），而对离群点进行挖掘分析之前，需要从中区分出真正的“离群数据”，而非“垃圾数据”。

异常检测根据原始数据集的不同可分为两类：

☐ **新奇检测（Novelty Detection）**：新奇检测的前提是已知训练数据集是“纯净”的，未被真正的“噪声”数据或真实的“离群点”污染，然后针对这些数据训练完成之后再对新的数据进行训练以寻找异常数据。

☐ **离群点检测（Outlier Detection）**：离群点检测的训练数据集则包含“离群点”数据，对这些数据训练完成之后再在新的数据集中寻找异常数据。

异常值检测常用于异常订单识别、风险客户预警、黄牛识别、贷款风险识别、欺诈检测、技术入侵等针对个体的分析场景。图 8-11 中白色点表示训练数据，深色点表示测试数据，两个椭圆区域表示“正常数据”区域，两条椭圆边界线表示“正常数据”和“异常数据”的边界，在边界之外的就是异常点的分布。

常用的异常检测方法可分为以下几类：

☐ **基于统计的异常检测方法**。该方法的基本步骤是对数据点进行建模，再以假定的模型（例如泊松分布、正态分布等）根据点的分布来确定是否异常。这种方法首先需要对数据的分布有所了解，进而通过数据变异指标来发现异常数据。常用的变异指标有极差、四分位数间距、均差、标准差、变异系数等。但是，基于统计的方法检测出来的异常点产生机制可能不唯一，而且它在很大程度上依赖于待挖掘的数据集是否满足某

种概率分布模型, 另外模型的参数、离群点的数目等都非常重要, 确定这些因素通常都比较困难。因此, 实际情况中算法的应用性和可移植性较差。

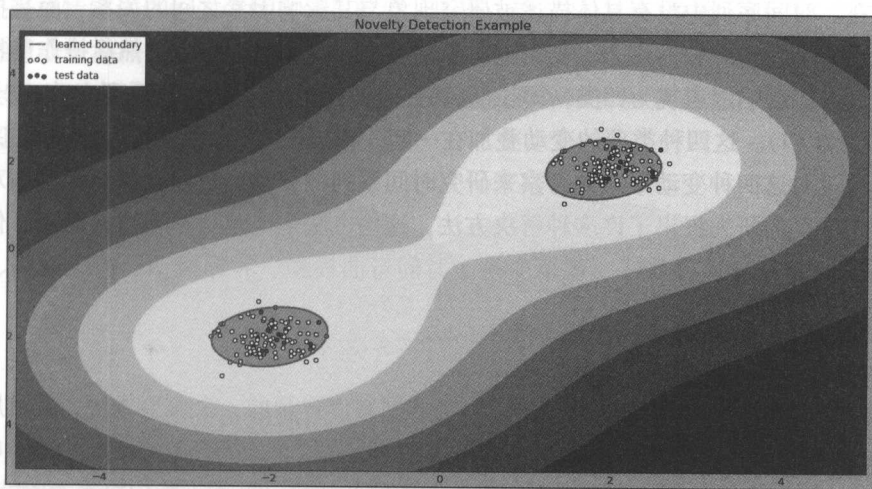


图 8-11 异常检测模型结果

- **基于距离的异常检测方法。**该方法定义包含并拓展了基于统计的思想, 即使数据集不满足任何特定分布模型, 它仍能有效地发现离群点, 特别是当空间维数比较高时, 算法的效率比基于密度的方法要高得多。算法具体实现时, 首先给出记录数据点间的距离 (例如曼哈顿距离、欧氏距离等), 然后对数据进行一定的预处理以后就可以根据距离的定义来检测异常值。比如, 基于 K-Means 的聚类可以将离每个类中心点最远或者不属于任何一个类的数据点提取出来而发现异常值。基于距离的离群检测方法不需要用户拥有任何领域知识且具有比较直观的意义, 算法比较容易理解, 因此在实际中应用得比较多。
- **基于密度的离群检测方法。**这种方法一般都建立在距离的基础上, 其主要思想是将数据点之间的距离和某一范围内的数据数这两个参数结合起来, 从而得到“密度”的概念, 然后根据密度判定记录是否为离群点。比如, LOF (局部异常因子) 就是用于识别基于密度的局部异常值的算法。离群点被定义为相对于全局的局部离群点, 这与传统异常点的定义不同, 异常点不再是一个二值属性 (要么是异常点, 要么是正常点, 实际上的定义类似于 98% 的可能性是一个异常点), 它摒弃了以前所有的异常定义中非此即彼的绝对异常观念, 更加符合现实生活中的应用; 但其缺点就是它只对数值数据有效。
- **基于偏移的异常点检测方法。**基于偏移的离群检测算法 (Deviation-based Outlier Detection) 通过对测试数据集主要特征的检验来发现离群点。目前, 基于偏移的检测算法大多都停留在理论研究上, 实际应用比较少。
- **基于时间序列的异常点监测方法。**所谓时间序列就是将某一指标在不同时间上的数值, 按照时间先后顺序排序而成的数列。这种数列虽然由于受到各种偶然因素的影响

而表现出某种随机性，不可能完全准确地用历史值来预测将来，但是前后时刻的数值或数据点的相关性往往呈现某种趋势性或周期性变化，这是时间序列挖掘的可行性之所在。时间序列中没有具体描述被研究现象与其影响因素之间的关系，而是把各影响因素分别看做一种作用力，被研究对象的时间序列则看成合力；然后按作用特点和影响效果将影响因素规为四类，即长期趋势（T）、季节变动（S）、循环变动（C）和不规则变动（I）。这四种类项的变动叠加在一起，形成了实际观测到的时间序列，因而可以通过对这四种变动形式的考察来研究时间系列的变动。目前，国际和国内对时间序列相似度的研究提出了许多种解决方法，这些方法主要包括基于直接距离、傅里叶变换、ARMA 模型参数法、规范变换、时间弯曲模型、界标模型、神经网络、小波变换、规则推导等。

8. 协同过滤

协同过滤（Collaborative Filtering, CF）是利用集体智慧的一个典型方法，常被用于分辨特定对象（通常是人）可能感兴趣的项目（项目可能是商品、资讯、书籍、音乐、帖子等），这些感兴趣的内容来源于其他类似人群的兴趣和爱好，然后被作为推荐内容推荐给特定对象。

协同过滤主要解决的问题是当客户进入某个领域后，什么内容或项目是他/她可能感兴趣的东西，然后以用户的兴趣为出发点推荐他/她可能感兴趣的内容，以此来提高用户体验、用户交互频率、订单转化效果、销售利润等。

协同过滤目前主要用于电子商务网站、兴趣部落网站、知识型网站、话题型网站、社交型网站的个性化项目推荐。协同过滤推荐的场景通常发生在，当客户对内容进行打分的前提下，比如内容评分、综合评价等。比如，当我在豆瓣音乐中将“范特西”专辑加入看过列表，并为它评分后，豆瓣又为我推荐了更多音乐人。



注意 协同过滤属于个性化推荐算法的一种，并不是个性化推荐的全部，除了协同过滤算法外，还有多种算法可以在不同场景下满足不同的推荐需求，例如关联算法、KNN、TOPN、深度学习等算法。另外，推荐算法也只是推荐系统中的一部分，除了推荐算法模块外，还可能包括实时数据接入和流处理，DMP 数据管理、用户行为建模和画像标签库、项目标签库、冷启动场景规则、异常场景规则、强制人工干预规则、模型和数据修正模块、场景引擎模块、投放引擎模块、机器学习算法库和挖掘模块、推荐规则模块、推荐效果分析模块等。

协同过滤常用的算法主要分为四类：

- ❑ 基于用户的协同过滤（User-based Collaborative Filtering）；
- ❑ 基于项目的协同过滤（Item-based Collaborative Filtering）；
- ❑ 基于模型的协同过滤推荐（Model-based Collaborative Filtering Recommendation）；
- ❑ 混合协同过滤。

9. 主题模型

主题模型 (Topic Model), 是提炼出文字中隐含主题的一种建模方法。在统计学中, 主题就是词汇表或特定词语的词语概率分布模型。所谓主题, 是文字 (文章、话语、句子) 所表达的中心思想或核心概念。比如, 当提到 IBM 时, 可能我们会想到 ThinkPad; 提到比尔盖茨, 我们就会想到 Windows。IBM 和 ThinkPad、比尔盖茨和 Windows 就是各自主题中相关的概念。

如图 8-12 所示为有关数据工作的主题。

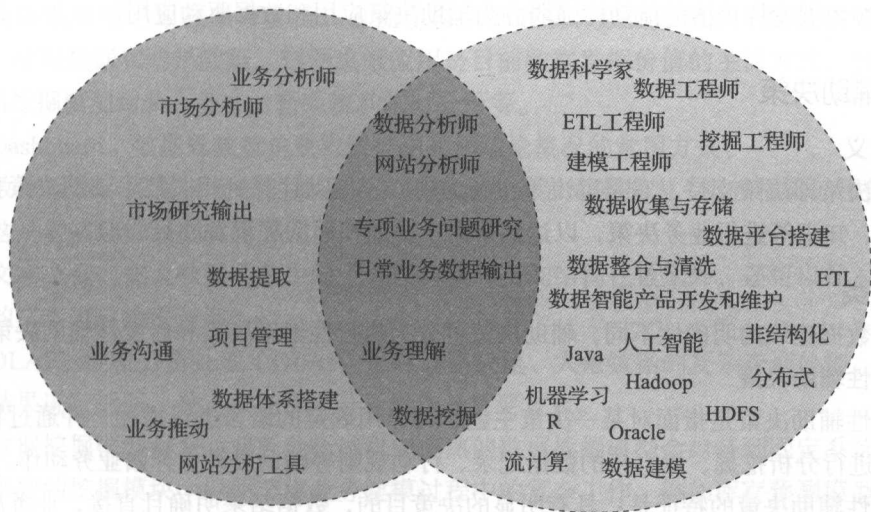


图 8-12 数据工作主题

主题模型是一个能够挖掘语言背后隐含信息的利器, 是语义挖掘、自然语言理解、文本解析和文本分析、信息检索的重要组成部分。

- 它可以衡量文档之间的语义相似性, 是文本聚类、分类、情感分析、文档相似度等应用的重要组成部分。
- 它可以解决多义词的问题, 实现准确的词性标注。
- 它可以排除文本中的噪声, 从中准确地提炼出主题关键字。

主题模型克服了传统信息检索中文档相似度计算方法的缺点, 并且能够在海量互联网数据中自动找出文字间的语义主题。主题模型可以应用到围绕主题产生的应用场景中, 例如搜索引擎领域、情感分析、舆情监控、个性化推荐、社交分析等, 主题模型得到的结果, 可以在去停用词之后, 配合标签云等形式做进一步的形象展示。

常用的主题模型包括:

- 潜在狄利克雷分配模型 (Latent Dirichlet Allocation, LDA);
- 概率潜在语义分析 (Probabilistic Latent Semantic Analysis, PLSA);
- 其他基于 LDA 的衍生模型, 例如 Twitter LDA、TimeUserLDA、ATM、Labeled-LDA、MaxEnt-LDA 等。



主题模型中的潜在狄利克雷分配模型跟线性判别分析的缩写相同，都是 LDA，实际上二者毫无关联，仅仅是缩写相同而已。

8.5 数据应用

数据应用是数据产生价值的出口，前期所有的数据动作都为后期的价值输出做准备。数据应用层按数据发挥价值的能动性强弱分为辅助决策应用和数据驱动应用。

8.5.1 辅助决策

1. 含义

辅助决策即决策支持，它是以决策主题为中心，借助计算机相关技术辅助决策者通过数据、模型、知识等进行业务决策，以达到帮助、协助和辅助决策者的目的。

2. 分类

根据数据建议的明确性不同，辅助决策可分为执行性辅助决策和启发性辅助决策。

执行性辅助决策

执行性辅助决策是指面对某一决策主题，可以用确定的语言进行描述，并通过特定的方法和模型进行分析挖掘，以直接的数据记录、行动规则等辅助决策方开始业务动作。

执行性辅助决策的特征是：具有明显的决策目的，数据结果明确且直接，业务决策方可直接采用其结果并落地到业务执行。

举例：某业务需要针对部分客户进行大型活动以实现促销，此时需要确定促销客户名单，如果数据从业者提供了客户名单、样本抽取规则等，可以直接帮助业务确定发送对象。

启发性辅助决策

启发性辅助决策相对于执行性辅助决策而言，可能没有面对某一决策主题，也可能是面对某一决策主题时没有明确的结果论断，但提供了间接的数据相关论证、规则、描述等，需要业务自身根据这些信息进行自我判断和决策。

启发性辅助决策的特征是：决策主题不明确或在明确的决策主题下没有明确的业务落地地点，决策方无法直接开展业务活动。

举例：日常的统计性数据报告、面向市场研究类的宏观报告都属于此类范畴。

3. 步骤

辅助决策的步骤通常分为 4 步：

1) 建立决策主题。业务方基于需求或问题形成决策主题，包括问题组成、决策方向、决策方法、实施周期、效果评测等，这是决策活动的起点。

2) 分析决策主题。数据从业者利用相关数据知识、工具、技能来定性分析和挖掘决策主题，并得出可供决策方应用的描述或结果。

3) 评估决策建议。决策方根据个人才能、经验、流程以及所处环境条件等因素对描述或结果进行评估,从而确定最优方案。

4) 决策实施。决策方落地决策建议并开展业务动作。

辅助决策往往不是一次性工作,而是一个迭代优化的过程。每一次决策实施既是上一次辅助决策的终点又是下一次辅助决策的开始。

4. 形式

辅助决策应用的数据来源广泛,可能包含数据源层、数据处理层、数据计算层等的原始数据、过程数据和结果数据。辅助决策应用是目前数据发挥价值的主流方式,包括报表支持、数据挖掘模型封装、业务分析系统和临时分析等。

- Dashboard。数据驾驶舱是获得数据洞察和结论概况的常用方式。
- 数据报表。数据报表根据数据结果的延迟性分为即席报表(实时报表)和普通报表,这是最基础的数据应用形式。
- 文本查询。在大数据平台中除了可以检索和查询结构化数据外,还可以针对非结构化的文本进行检索查询。
- OLAP。联机分析处理(OLAP)可以提供快速、大量数据的复杂查询处理,并以查询结果进行返回,是复杂场景下分析操作的重要方式。
- 数据挖掘模型封装。业务分析过程中成熟的数据挖掘模型会封装到固定系统内,形成单独的挖掘模块,以便节省业务建模过程中的复杂工作,例如库存预测模型、流失预警模型等。
- 业务分析系统。业务分析系统包括流量分析系统、客户分析系统、销售分析系统等,分析系统是相对完善的分析体系,相较于报表具备更多分析和挖掘类功能。

8.5.2 数据驱动

1. 含义

数据驱动是指整个业务运作流程以数据结果为运作目标,以关键数据为触发方式,借助计算机相关技术结合企业内部流程和机制形成数据一体化工作流程。

2. 分类

按照自动化程度不同,数据驱动分为自动化数据驱动及半自动化数据驱动。

自动化数据驱动

自动化数据驱动指数据业务流程,从确定决策目标、决策分析、决策评估到决策执行过程都是完全自动化的驱动方式。自动化数据驱动的核心是整个过程除了人为调参优化或加入人工干预规则之外,不需要借助其他业务方的参与。

举例:常见的站内个性化推荐系统是自动化数据驱动的典型应用,站内个性化推荐系统实现了数据自我决策实施的整个过程,除了调优和人工干预规则外,无需人工介入执行。

半自动化数据驱动

相对于自动化驱动过程，半自动化数据驱动需要人工介入，主要介入点是人工代替机器或系统触发业务动作，但业务方的角色是仅作为实施方介入，不参与决策过程。

举例：大多数电子商务网站都有基于加入购物车事件的触发机制，当登录或注册的用户将商品加入购物车但放弃购买后，自动会发送短信或邮件提醒用户继续购买，并附以折扣、限时、优惠券等措施刺激用户完成订单。部分公司由于整个系统并未完全打通，因此其中的客户名单需要人工梳理后录入发送平台，进而完成整个提示过程。

3. 步骤

数据驱动的步骤与辅助决策相同，只不过在建立决策主题、分析决策主题、评估决策建议和决策实施过程中都是由数据配合开发的自动化系统完成，整个决策的载体是自动化数据+业务系统（IT 是载体），核心是数据本身。

4. 形式

目前，数据驱动应用的数据大多来源于计算层的实时计算和临时计算结果，对于时间性要求不强的场景主要调用离线计算数据结果。

数据驱动需要借助技术手段实现，通常是建立在数据事件触发或数据结果触发基础上的自动化运行机制。

- 智能广告类：包括 RTB、个性化 EDM、程序化购买、精准营销系统等。
- 智能网站管理类：包括站内个性化推荐系统、个性化着陆页、智能资源循环与利用、基于用户事件或时间的触发等。
- 智能客服类：自动问答系统、基于语言的自助业务办理系统、智能呼叫、聊天机器人、自动知识管理等。
- 智能调度类：智能调价、智能资源分配、智能负荷管理、最优规划管理、动态优化配比等。
- 智能工厂类：智能化生产控制、智能化执行过程控制、智能生产线、智能仓储和物流运输体系、智能设备检测与维护、智能环保管理等。
- 智能生活类：包括智能家居、智能家电、智能穿戴、智能移动、智能社交、智能购物、智能办公等。
- 其他智能应用：智能交通、自动驾驶、智能会议系统、机器人等。



注意

辅助决策和数据驱动是两个层次的数据应用，数据驱动相对于辅助决策的实现难度更高、数据价值体现更大。但很多时候往往难以分清楚二者的区别，二者的差异点在于辅助决策为业务决策方服务，整个过程都由业务人员掌控，数据是依存和辅助的角色；而数据驱动的过程由数据掌控，数据是主体，实现该过程需要自动化系统、算法等支持，因此，数据驱动具有自主导向性、自我驱动性和效果导向性的特征。由于数据本身会存在缺陷以及业务需求，需要在数据运作过程中加入人工干预因素。但数据作为数据驱动的核心不变，数据即决策本身。

8.6 数据质量管理

随着数据类型、数据来源的不断丰富以及数据数量的快速增长,企业在数据管理工作和数据流程中面临着越来越多的数据质量问题,例如多库表间的数据不一致、数据质量差、ETL 和数据整合难度大、数据技术工作缺乏统一标准等,这些问题已经严重影响数据后期业务应用的效果甚至辅助决策的正确性。因此,数据质量把控与建设的重要性逐渐进入企业高层的关注视野。

8.6.1 数据质量建设的内涵


数据质量问题可理解为有没有数据、数据能不能用、数据好不好用、数据如何用的问题。数据质量建设是对数据规划、产生、存储、分析、应用、维护整个流程中可能产生的问题进行预测、监控、识别、处理的过程,其内涵包括数据和数据工作流程的规范化、标准化、流程化,目的是提高企业数据的应用效率和实际效果。

大多数企业的数据质量建设都是 IT 部门的工作,而 IT 部门通常把数据清洗或 ETL 的过程理解为数据质量建设。数据清洗和 ETL 是数据质量建设的重要过程,主要内容是针对原始数据中的错误值、异常值、缺失值等进行处理,另外通过转换规则生成新的数据。

但是,数据清洗或 ETL 只是数据质量建设的一环。数据质量建设从数据规划开始,贯穿数据从诞生到消亡的整个过程,它是一个周而复始的循环。另外,数据质量建设也不仅仅是针对技术体系的规范,而是需要约束所有参与数据工作的技术体系和业务体系。

不同企业对于数据质量的关注程度不同。通常而言,大企业由于自身数据量级的增加以及应用场景的丰富,对数据的认识会更全面,因此更加重视数据质量。小企业由于数据规模较小以及业务需求不大而对数据的体会较少;另外,大多数情况下,数据对初创企业的直接利润贡献较小,因此不会被企业过多重视。

数据质量建设需要从企业初期开始规划并逐步实施,而不是等到企业做大之后;否则数据会由于迁移成本高、数据类型混乱、数据整合难度大、丢失关键数据信息等导致永久性数据质量的问题。

 **注意** 数据的生产环境具有实时性特征。由于数据采集问题导致的数据缺失、值错误、丢失记录等问题是无法通过后期处理还原的,因此数据的不可逆问题将会严重影响数据质量。

无论是大企业还是小企业,数据质量管理都没有得到其应有的重视程度,根本原因体现在数据成本和产出的关系上:

□ 数据质量管理成本高。数据质量管理过程涉及企业数据标准的规划和制定、数据规范的落实监督、数据生命周期等,数据质量工作的每个环节都需要大量的人力、物力、财力和时间成本。

- ❑ 数据质量的效益和结果不明显。大多数企业都没有把数据质量考核纳入 KPI 考核体系中，即使出现数据问题也可以在后期采用多种处理方式“蒙混过关”，这些无法被企业高层发现，尤其对于缺乏数据文化的企业来讲更是大事化小、小事化了。

事实上，大多数公司对于数据质量的认知只停留在“有数据”的层次，由于缺少完善的数据工作标准和规范、数据应用的绩效管理和数据效果的验证，常常导致各种数据应用问题：

- ❑ 不同系统出具的企业关键指标不一致，例如销售金额、订单量、利润；
- ❑ 当企业需要实时数据做辅助决策时，发现没有数据可以使用；
- ❑ 企业内部不同部门间的数据无法共享，并且沟通成本非常高；
- ❑ 领导在听取各部门工作汇报时，发现每个部门都在汇报相同的指标但结果都不相同；
- ❑ 由于数据工作时效性特征明显，数据工作者在数据清洗和处理上花费了大量时间，相对地只能减少在深入分析和挖掘上的精力投入，最终产出价值和驱动效果有限；
- ❑ 企业内部存在非常多的数据孤岛，数据之间无法关联且冗余明显，IT 部门日常数据维护需要投入很多软件、硬件、时间和人力成本。

以上问题的长时间累积会导致数据工作的价值逐渐缩小，企业对于数据的认可度、信任度、依赖度降低，数据工作将面临被企业边缘化甚至裁撤的风险。

8.6.2 影响数据质量的常见因素

1. 数据标准管理影响

数据标准管理也可以理解为元数据管理，元数据是关于数据的数据，即把数据作为一项主体对待，通过数据对数据主体进行定义、规范、描述和记录。

数据标准作用于数据工作的完整生命周期，如果没有有效的数据标准管理，将导致以下问题：

- ❑ 不同的开发人员有不同的开发标准，数据开发质量无法保证；
- ❑ DBA 在数据管理过程中，由于缺乏维护标准而导致数据的丢失；
- ❑ ETL 人员依靠自身对需求的理解进行数据抽取、转换和加载，导致结果和需求不一致；
- ❑ 原有的数据管理人员离职，新入职人员由于没有数据标准可供参考而无法有效进行数据维护，甚至由于对库表定义的不确定而无法提取有效数据；
- ❑ 相同数据主体在不同库表间的数据差异较大；
- ❑ 业务人员无法准确理解数据指标的含义，数据应用无从谈起；
- ❑ 业务人员过多的基本定义问题直接影响了技术人员的工作效率和效果。

造成以上问题的根本原因可归结为三类：

- ❑ 第一类是缺少数据标准，即没有元数据规范；
- ❑ 第二类是数据标准不完整，完整的数据既包括数据主体又包括数据流程，大多数企业缺少对数据工作流程的规范，例如协同工作流程、复合指标定义规则、业务应用规则等；

□ 第三类是数据标准更新不及时，数据标准同样需要与时俱进，需要根据企业发展的不同阶段及业务的不同侧重点进行调整。

2. 数据客观环境影响

数据客观环境既包括数据工作所依赖的机房、硬件、软件、程序等工作环境，还包括互联网等企业外部大环境，这些环境的变化也会导致数据质量的下降。例如：

- 机房由于停电、被盗、洪水等事故或不可抗力因素造成数据丢失或损害；
- 硬件由于使用寿命等因素造成了数据物理损害；
- 软件和程序的漏洞、BUG 甚至文件系统错误导致的数据系统无法正常工作；
- 互联网访问的限制（例如禁止某些网站的访问和使用）、变更或黑客入侵对数据的恶意篡改、删除等因素。

数据客观环境的影响大多是由客观不可控因素造成的，但其中也不排除某些人为疏忽和工作漏洞因素，例如数据备份管理和灾难性恢复机制的缺失、企业网络高危漏洞的监控和修复缺失等。

3. 工作流程管理影响

数据应用流程管理指在数据工作过程中对数据工作流程、日常维护、数据使用的管理，常见的应用流程管理不规范的场景包括：

- 数据需求和开发信息不对称，导致数据理解、数据范围、数据精度、数据粒度上出现偏差，或者在数据开发过程中不断变更需求而影响开发进度；
- 缺乏数据项目工作效果的监督，导致数据工作的各个环节质量难以保障；
- 数据维护和使用规范不全面，缺乏统一的落地标准；
- 缺少立体监督机制，数据部门既是运动员又是裁判员，企业难以全面把控数据质量和风险。

在数据工作过程中，除了技术人员外，业务人员或企业职能人员的参与也会影响数据质量。因此，数据质量建设过程中需要不同层级、不同环节的各个部门都参与到数据工作标准和规范的制定中。

除了以上三方面的主要影响因素外，相关数据、技术和业务人员的自身意识、素质、经验和技能水平，企业内部的数据文化和落地机制，企业发展阶段和其他客观环境也都会影响数据质量，但这些因素更多地属于数据工作不可控的范畴。比如，企业在发展前期会重点关注业务工作流程的规范和业务效果，而对于数据工作范畴的投入和重视程度较低，客观上会导致初期数据质量较低。

8.6.3 数据质量建设的框架

数据质量建设需要从数据工作的整个流程入手，通过全方位、立体化、动态性的标准建立完善的监督和管理体系，如图 8-13 所示。

数据工作流程可笼统地分为数据层、处理层和应用层：

- ❑ 数据层包括生产环境的原始数据采集、数据 ETL 及数据整合和存储；
- ❑ 处理层包括数据的分析、挖掘和建模处理过程；
- ❑ 应用层包括数据产品应用、数据报表、数据接口和第三方应用等。

数据质量建设的框架可分为数据质量管理、数据监督管理和数据生命周期管理三方面：

- ❑ 数据质量管理：数据整个工作流程所有环节的标准制定和规范管理；
- ❑ 数据监督管理：通过数据对数据主体进行监控及数据质量的分析和改进；
- ❑ 数据生命周期管理：动态更新和管理数据各个环节的标准和规范，使得数据主体、数据质量和规范、数据监督管理不断优化并符合实际情况。

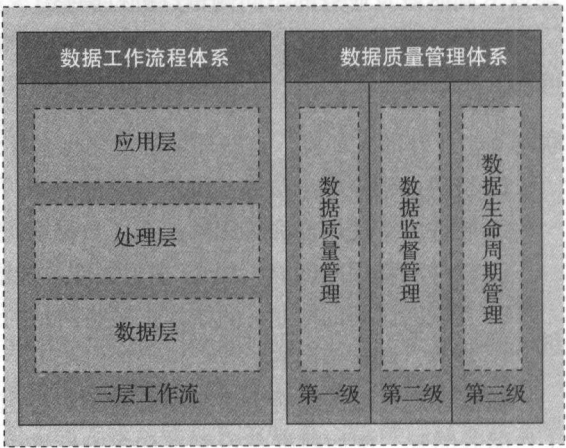


图 8-13 数据质量建设的框架

1. 数据质量管理

数据质量管理是数据工作的指导和规范文件，主要用于数据的开发、管理、维护、处理 and 应用的参照。根据流程可将数据工作标准分为数据开发标准、数据处理标准、数据存储标准、数据建模标准和数据应用标准，如图 8-14 所示。

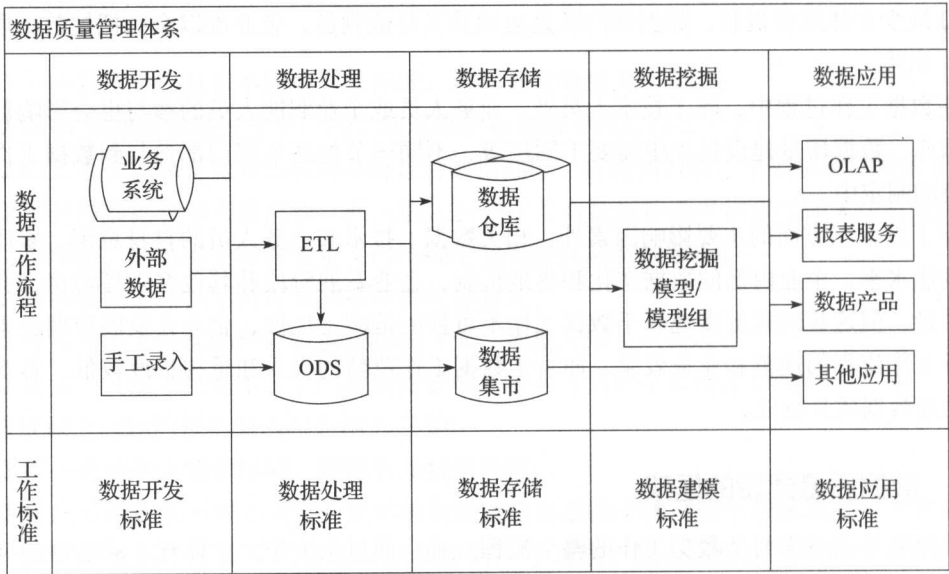



图 8-14 数据工作流程与数据质量管理体系

- ❑ 数据开发：通过业务环境采集获取原始数据、外部数据以及手工录入数据。
- ❑ 数据处理：原始数据通过 ETL 流程进行数据抽取、转换和加载，对于其中规则复杂的部分配合 ODS 区进行处理。
- ❑ 数据存储：经过数据处理规则后的数据形成数据仓库或数据集市。
- ❑ 数据挖掘：通过数据挖掘模型进行数据深入分析，之后会得到数据标签或模型结果而成为数据仓库或数据管理平台的主要内容；部分应用如汇总报表等不需要挖掘过程，因此会直接到达数据应用层。
- ❑ 数据应用：数据通过报表、数据门户、OLAP、数据产品等进行业务应用，应用数据可能来源于数据仓库、数据集市或数据挖掘模型。

 **提示** ODS 是一个面向主题的、集成的、可变的、当前的细节数据集，常用于向数据仓库的过渡方案。ODS 与数据仓库的区别在于其主题集中度介于分散数据与数据仓库之间，数据通常只有当前或近期的数据，支持数据的增、减或删除动作。ODS 可以在数据仓库与原始数据之间形成一条过渡地带，用于完成数据仓库之前的数据存储、处理或特殊应用。

数据开发标准

数据开发标准是针对数据采集过程的标准，不同的业务系统具有不同的开发标准和需求，数据开发标准涉及数据开发计划、测试、实施和上线四部分，具体以企业实际为准。

数据处理标准

数据处理是将源数据转换成目标数据的过程，按照其流程数据处理标准涉及数据处理设计、开发、测试和运维四个阶段。

(1) 设计

数据处理设计包括源数据、目标数据、处理规则等。

源数据和目标数据的标准涉及的内容包括库名、表名、字段名、约束条件、类型以及其他字段说明等信息。如表 8-6 所示为一目标表内容的一部分。

表 8-6 目标表标准规范

表名	TableName	字段（列）	说明
品牌表	DIM_BRAND	BRANDID	品牌 id
品牌表	DIM_BRAND	SOURCEID	来源
品牌表	DIM_BRAND	BRANDNAME	品牌中文名
品牌表	DIM_BRAND	BRANDENAME	品牌英文名
品牌表	DIM_BRAND	INSERTDT	插入时间
品牌表	DIM_BRAND	UPDATEDT	修改时间

数据处理规则主要涉及表间的关联和规律规则、转换规则、合并规则、匹配规则、触发机制以及其他复杂运算规则等。如表 8-7 所示为针对特转换规则进行的定义。

表 8-7 处理规则定义

函数名称	输入	输出	备 注
conv_ip	12.22.22.22	12220220220	ip 地址按小数点分成四个部分 part1 ~ part4 part1 × 1+part2 × 1000+part3 × 1 000 000+part4 × 1 000 000 000
conv_id	42060219820917101C		将 15 位身份证转化成 18 位身份证，将无效的 身份证信息去除
conv_mobilenumber	+8613831222222 013818888888888 001381888888888	13831222222 13888888888 13888888888	符合规则 +86 860 086 000 开头的后面紧跟 13 × 15 × 18 × 再后面紧跟 8 位数字的手机号码 截取 (13 15 18) \d{9, 9} 不符合规则的一律置 空 13 × 后面紧跟数字全部相同的一律置空

(2) 开发

数据处理开发过程中的标准通常涉及脚本语言和 SQL 语言两种。

脚本语言通用的标准管理内容包括存放目录、任务组、程序名、变量名、大小写、任务标记、脚本编号、公用调度的函数处理、模板标准等。除此以外，根据结构可分为程序描述区、全局变量定义区、主程序体三部分。

- ❑ 程序描述：脚本创建日期、创建人和修改记录。
- ❑ 全局变量定义：变量显示声明、通用模块的引用、全局变量定义，具体包括引用顺序、初始值声明、注释强制性、对齐和缩进要求等。
- ❑ 主程序体：函数组成、复用说明、对齐和缩进数、数据读取规则、赋值规则、调用规则和退出规则等。

SQL 编写标准涉及的内容包括库、表、存储过程、宏、字段名等名称的大小写，同级或不同级别 SQL 的缩进和换行，特殊符号位置（例如逗号位于行首而分号位于行末），别名使用唯一性，空格使用和变量引用的规范性，注释的完整性要求尤其是其中特殊处理的说明，内连接和外连接的固定用法及配合筛选条件的使用规范、空行的使用以及超长脚本的注意事项等。

(3) 测试

在数据处理开发完成后，通常需要经过多轮测试成功后才能上线；对于测试的流程、规范性验证、性能要求、触发条件等都需要制定完整的标准。

(4) 运维

数据处理调度作业列表顺序、时间，目标数据更新规则，版本控制，错误处理流程（根据抽取、转换和加载的不同阶段进行细分）、调度异常处理机制、日志检查机制等，运营维护文档也需要有统一的规范。

数据存储标准

数据存储标准涉及数据仓库设计、数据仓库开发和数据仓库运维三部分。

(1) 数据仓库设计

数据仓库在实际开发之前都会进行设计，设计阶段的主要内容包括：

- ❑ 数据仓库层次结构设计：STAGE 接口信息模型、ODS/DWD 信息模型、MID 信息模型、

DM 信息模型、元数据信息模型的规范；

- ❑ 数据仓库设计：概念模型、逻辑模型和物理模型的流程标准，不同库、表和分区的划分依据以及应用场景，库表容量大小限制和增长设计，范式参照依据及优化原则等；
- ❑ 分级存储设计：根据数据的访问频率、重要性、保留时间、数据量等进行层级化存储，包括在线存储、近线存储和离线存储规则。

(2) 数据仓库开发

- ❑ 数据库对象命名：不同类型表的前后缀规范，例如维度表、事实表、接口表、汇总表、临时表等；主键、外键、索引、视图、物化视图、存储过程等命名标准。
- ❑ SQL 编码规范：除 ETL 中的 SQL 要求外，对于全表扫描、删除、排序、编码、字符长度、SQL 语句优化和性能控制等都需要进行明确规范。
- ❑ 数据字典：数据字典是数据库开发和实施的重要参考依据，数据字典是指对数据的数据项、数据结构、数据流、数据存储、处理逻辑等进行定义和描述。如表 8-8 所示为流量据仓库数据字典的一部分。

表 8-8 流量数据仓库数据字典示例

表名称	列名称	列说明	数据类型	键值	示例 / 对照
DIM_REPORT_SUITES	USER_HASH	有关报表包 ID 的散列	int (10) unsigned		3757249664
	USERNAME	报表包 ID	varchar (40)		omniture
	USERID	报表包 ID 的数字 ID (通常可以在 DW 文件名中看到)	int (10) unsigned		644614
FACT_VISITOR_ENVIRONMENT	VISITOR_ID	由 visid_high 和 visid_low 组成	bigint(100) unsigned	FK	
	VISITOR_ID	由 visid_high 和 visid_low 组成	bigint(100) unsigned	PK	11111112-14753
	USER_ID	用户 ID	varchar(255)	FK	32265
	DATE_TIME	报表包 ID 指定的时区中的时间 (可读格式)。由 Adobe 服务	datetime		38261.00141
	ACCEPT_LANGUAGE	浏览器中的接受语言标题	varchar(20)		en-us, ja: q=0.62, de
	C_COLOR	从 JavaScript 生成的颜色深度	varchar(20)		32
	CODE_VER	创建图像请求的 JS 文件中的代码版本	varchar(16)		G.7-pD-S
	COOKIES	是否接受 Javascript 会话 Cookie	char(1)		Y
	PERSISTENT_COOKIE	表示是否已启用第三方 Cookie 和 / 或永久性 Cookie 的标记	char(1)		Y
	CT_CONNECT_TYPE	浏览器计算机的连接类型	varchar(20)		调制解调器
	EVENT_LIST	代表从客户处传递的事件的数字 ID 列表 (以逗号分隔)	text	FK	请参见事件对照

(3) 数据仓库运维

设计仓库运维过程中对于日志、异常处理、并行支持、循环作业、调度流程等进行规范。



提示 数据字典与元数据的区别：通常而言，数据字典是元数据的一部分；数据字段主要对数据进行描述，例如属性、约束、含义、解释等；而元数据的范围更广，除了对数据自身静态的描述外，还包括对数据流程、数据处理等的定义。

数据建模标准

数据计算涉及不同的场景和应用需求，每种场景和需求对应的计算工作标准都是不同的。这里以数据挖掘流程的 CRISP-DM 标准为例说明整个建模过程。CRISP-DM 模型是目前数据挖掘领域应用最广泛、接受程度最高的挖掘流程标准，该标准将数据挖掘过程分六个步骤，如图 8-15 所示。

(1) 商业理解

商业理解是从商业角度理解挖掘项目的目标和要求，然后把理解结果转化为数据问题，并制定出一个旨在实现目标的初步计划。这个环节中对于商业理解的目标、要求、内涵、范围、时间、结果类型以及结果细节都要做明确的规范。

(2) 数据理解

数据理解是从原始数据集开始熟悉和了解数据，并初步探索数据知识，或挖掘有深层含义的数据子集来形成对隐藏信息的假设。数据理解可以理解为数据的可行性研究，即通过先期的基本数据认识来确定当前数据条件是否可以满足数据挖掘所需条件并初步判断如何实现的问题。

数据理解阶段，需要确定源数据标准如数据来源、范围、状态或时间性特征，数据集基本特征如数据类型、最大值、最小值、均值、标准差、偏度、唯一性、有效记录数及数据分布规律等。

(3) 数据准备

数据准备是从原始数据集中按照数据理解的要求和规则进行数据处理的过程，这个过程的主要工作是数据清洗，为下一阶段数据建模准备数据。

这个环节需要规范的内容包括数据质量修复策略如极值、异常值和缺失值，确定抽样方法以及数据处理过程中涉及的数据类型、格式、条件等转换规则。

(4) 建模

在建模阶段，主要是选择各种建模技术，同时对参数进行校准以达到最优拟合状态或输

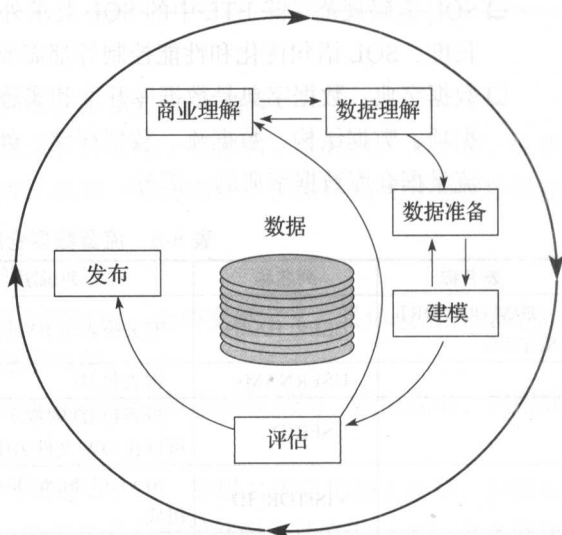


图 8-15 CRISP-DM 流程

出理想挖掘结果。数据建模阶段是整个数据挖掘的核心，需要重点关注并规范以下几个方面：

- 应用场景和算法类型规范，不同应用场景下如何选择挖掘算法的类型及优劣注释。
- 不同算法类型及算法细化，例如同样的关联规则细分到交叉销售和向上销售各自的算法。
- 参数调优方法，不同参数在面对不同类型、不同体量、不同场景和不同目标下如何调优，以及产生的预期效果。

提示 从数据挖掘的应用需求看，参数调优是最具有知识含量也是最需要指导和规范的环节，对于这部分内容的标准化不仅可以提高数据挖掘效果，而且更有利于数据挖掘知识的沉淀。

(5) 评估

评估阶段的目的是保证数据挖掘和拟合结果符合数据实现逻辑并能实现业务目标。评估是发布或商业应用前的最后一个质量把关环节，通过三个标准进行度量：

- 用户的满意程度。只针对模型结果的应用对象是“人”，即业务人员是才应用的度量，实现业务需求是数据挖掘结果的基本要求。
- 数据模型质量评估。数据挖掘模型需要通过一定的指标评估其拟合程度、显著程度或规则的可用度。不同模型有不同的评估指标，例如关联模型的支持度、置信度，回归模型中的解释系数、T值和P值等。如图8-16所示为决策树提升效果图，图中显了不同样本比例下的提升幅度。

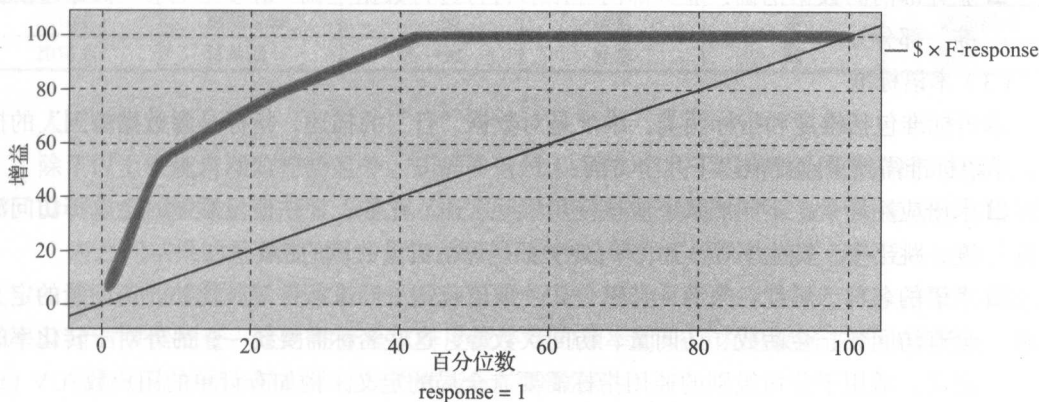


图 8-16 决策树提升效果图

- 技术可用性评估。对于需要集成到其他应用或产品中的挖掘模型，还需要从技术上评估其可用性，例如规则应用效率、并发处理量、响应时间等。

(6) 发布

挖掘模型的发布意味着数据建模进入到实际应用或部署阶段。数据模型的发布既是一个

模型的阶段性结束，又是下一次调整的开始。模型优化是一个循序渐进的过程。这个阶段需要重点关注发布或部署流程细则，包括部署时间、应用范围、应用条件等。

数据应用标准

数据应用标准从实际应用的角度描述了所要应用的数据或系统的基本特征，提供了业务应用所需的来源、规则、定义和属性等信息，数据应用标准分为流程标准、场景标准和术语标准三个层面。

(1) 流程标准

流程标准的核心是定义数据与业务间的沟通和协作机制，包括需求处理流程、协同测试流程、产品开发参与流程等，流程规范是保证各方有序参与并高效率协作的必要保证。

(2) 场景标准

场景标准主要规范不同场景下如何应用不同的数据集或数据结果，大多数企业的应用场景可分为报表应用、OLAP、数据库操作、业务部门的数据挖掘等。

❑ 报表应用：定义不同场景下查询报表的规范，包括库和表查看范围、权限范围（编辑、下载、打印等）、指标范围、数据集范围等。

❑ OLAP：不同需求下选择和应用数据维度及指标的规范，尤其是避免细粒度、长时间范围内的数据调用。对于常用的场景、维度和指标的组合可提供建议规则，提高业务效率的同时也能提高数据产出效果。

❑ 数据库操作：包括数据库操作的数据范围、访问时间、可编辑权限、验证性访问规范、SQL 语言规范，重点是通过标准来避免多表关联时的低效率以及对 IO 的过度占用。

❑ 业务部门的数据挖掘：业务部门导出后自行进行数据挖掘（请参见上节“数据建模标准”部分）。

(3) 术语标准

术语标准包括维度和指标两类，维度是对数据“行”的描述，指标是对数据“列”的描述。术语标准需要重点规范以下几个方面：

❑ 术语应用场景，不同场景下应该使用哪些术语。比如，评估流量质量时会选择访问深度、跳出率、转化率等，而 UV、PV 都是评估流量数量的指标。

❑ 术语的名称、属性、来源、计算公式、取值范围、约束条件等。比如，访问量的定义会有访问数、会话数、访问量、访问次数等，这些名称需要统一；另外对于转化率的定义，适用于公司级别的通用指标需要有全局的定义，例如有订单的用户数 /UV（或访问量）；而各个部门做内部分析时可根据情况制定特有指标，例如销量 /UV 等。



注意

规范和定义术语标准是企业内部数据流通和共享的唯一保证，也是内部有效应用数据的基础。如果没有这个过程，企业内部各部门之间无法进行有效的数据交流。比如，同样一个销售额，财务、销售、营销、物流部门的数据都会不一致。

2. 数据监督管理

数据监督管理是对数据本身以及数据工作流程进行监督和管理的过程，通过实时监控与预警、人工信息校验和逐级审核机制三方面综合实现。

实时监控与预警

实时监控是针对企业数据系统的动态进行实时监控，以及时发现数据的不完整、不一致、不及时和不准确等异常，并以邮件、短信或桌面通知等多种方式智能预警，帮助企业达到维护数据质量的目的。数据监控包括以下几方面内容：

- 数据库工作环境的监控：除了 IT 系统运维所要监控的服务器硬件和软件信息外，还包括服务器数据库表空间及使用率、基本配置监视、时间文件性能等。
- 数据库工作进程的监控：数据库事务处理、数据缓冲区、数据库连接时间、请求统计、连接统计、线程统计、缓冲统计、数据库明细、表锁统计、会话量、会话等待、SQL 占用资源、链效率、备份明细、计划 Jobs 等。
- 数据结果监控：在终端数据结果中，对关键指标的数据异常、涨跌的监控。
- 特殊事件监控：比如，黑客入侵、关键表的错误删除等的监控。

以上监控通常需要通过系统检测工具或脚本触发实现，并且可以通过设置一定的阈值与触发方式配合预警后的关键落地动作。如表 8-9 所示为某天针对关键指标的系统汇总的预警信息。

表 8-9 针对关键指标的预警信息

日期	指标	一致性	波动性	警告级别	警告方式
2016.8.1	销售总额	一致	正常	无	无
2014.8.1	会员总数	不一致	正常	低	邮件
2016.8.1	利润总额	不一致	异常	中	桌面提示
2016.8.1	订单量	不一致	异常	高	短信

人工信息校验

除了以上系统自动监控信息外，还需要通过人工参与进行关键数据信息监控，包括：

- 日常数据校验：通过汇总数据的记录数、不同指标的有效记录数、关键指标在不同库表之间结果的差异性对比、ETL 执行日志校验等方式确认数据是否违反一致性、及时性、完整性原则。日常校验的频率可分为日、周和月，校验数据范围依据频率而定。
- 定时数据抽查：定时抽查是以季度或半年的频率对期间内的数据进行抽样调查，选择重点库表并配合随机抽样及关键指标校验数据。
- 全面数据校验：每年组织一次针对所有数据仓库、集市和数据表的全面检查，包括所有维度和指标，目的是排除日常和抽样过程中的遗漏信息以及信息死角。

在人工校验时，还需要对数据总量的容量、增长速度、趋势走向，关键细分库表的变动趋势、结构构成、相关程度及重要变化进行统计，找到各个指标的波动范围标准或增长趋势规律，对带有影响因素的因子如节假日等还要加入特定权重进行调整。如表 8-10 所示为针对某指标制定的波动范围及阈值控制标准。

表 8-10 某关键指标的波动评估

指标 A	检查结果	波动上限	波动下限	状态
指标值	1.26%	2.15%	0.98%	正常
同比波动	2.3%	1.9%	3.1%	正常
环比波动	15.4%	21.3%	18.4%	异常

逐级审核机制

逐级审核机制是在数据应用过程中保证数据质量的重要关口，该机制的约束主要体现在两个场景。

- ❑ 数据变更时的逐级审核：在对数据进行插入、更新、删除、清空等关键操作时，必须建立逐级审核机制；尤其对于数据表级别影响范围较大或删除、清空等权限较高的操作，需要审批后方可执行。
- ❑ 数据汇报时的逐级审核：在得到数据结果后，数据应用部门首先要进行内部的逐级审核，以确保数据工作过程和结果没有问题，这个机制对于新入职的员工尤为重要。

3. 数据生命周期管理

数据生命周期管理的概念

数据生命周期从数据规划开始，包括开发、产生、部署应用、监视、存档和销毁，是一个不断循环的过程。数据生命周期管理是通过对数据不同阶段的针对性管理措施来降低数据管理成本并提高数据质量度，最终达到数据价值最大化的目的。

数据生命周期管理的意义

数据生命周期管理对于企业具有非常重要的意义，主要表现在以下几方面：

(1) 降低数据安全风险

数据在企业内存在损毁、泄露等显性风险，同时也存在由于数据生命周期导致的数据决策管理和数据驱动失误等隐性风险。比如，企业内部可能仍然使用早期的数据结论来辅助决策，由于时过境迁，早期的数据结论可能已经失去其存在或应用的条件，其结论的可信度需要重新评估，混乱应用将带来决策风险。

(2) 降低数据维护成本

IT 部门在面对越来越多的数据时，其运维过程需要投入大量的机房、硬件、软件、人力、物力和时间成本，这给企业造成了巨大的成本投入问题。数据生命周期通过各种措施，尤其是数据归档、销毁等方式有效进行数据管理，保证数据可用性的同时还可以有效降低运维成本。


(3) 提高数据质量度

数据生命周期管理的重要内容是针对数据本身的管理，通过对数据的开发、维护使得数据进入“去伪存真”的正向循环，越来越多的高质量数据的沉淀为后期应用提供了坚实基础。

提高数据应用价值

随着数据量的积累、数据类型的扩展等客观条件，以及企业在应用实践过程中数据经验

的丰富等主观条件的具备，数据生命周期管理还可以将隐藏在海量数据间的深层关系挖掘出来，最终为企业辅助决策甚至新商业模式的探索提供有效建议。

 **提示** 大多数企业关注数据生命周期管理的主要原因是降低数据运维成本和数据安全风险，但对于数据应用价值的关注和重视程度不高。数据存在的首要目的是辅助决策，再上一个层次是数据驱动，如果没有这两方面的效果数据将是企业的“负载产”，将没有存在的必要性。因此，数据生命周期管理的根本意义在于数据降低成本的同时提高效率，即“数据 ROI”的优化管理。

数据生命周期管理的实施

(1) 数据规划

数据规划是数据生命周期的开始，在规划阶段需要注意以下几个要素：

- **沟通主体**：数据、技术与业务共同参与并讨论数据采集规划，三方缺一不可，提高效率的同时还能降低沟通成本及后期调整成本。
- **沟通内容**：数据采集需求如数据主体、范围、标准、条件、粒度等，数据可行性讨论，预期产生的数据及后期应用方向，详细、具体的书面描述必不可少。
- **沟通结果**：经过业务、技术确认后的数据采集计划书，包括数据采集的所有内容和方法。

(2) 数据开发

数据开发主要是技术实施的过程，在开发测试阶段也会涉及数据或业务部门协助测试，测试过程的要点是全面、细心、准确，如果有 BUG 跟踪系统最好通过 BUG 跟踪系统跟进项目实施。

封闭测试也是数据开发过程中的重要内容，如果可外部访问注意数据的隐秘性及安全性，测试系统往往极易遭受外部攻击。

(3) 数据产生

数据开发完成后，生产数据开始正式进入数据库或日志系统，这是后期可供应用的原始数据。

(4) 数据部署应用

数据部署和应用包括数据 ETL、预处理、分析、挖掘、应用、流转和共享等，是数据产生价值的核心环节。需要注意的是，数据工作的对象以及每一个数据应用主体都有自己的生命周期，例如报告、模型、报表、产品等，这些主体都需要根据其自身生命周期进行调整，原因如下：

- 原始数据中出现了新类型的数据，这些数据可以用来拓展新的分析思路。
- 随着高质量数据的积累，由于数据量不足以产生可信结果的问题会逐渐解决，同时隐藏在深层次的数据间的关系会更容易被挖掘出来。
- 业务不同阶段有不同的关注点，数据结果需要随着业务的关注点而不断演进。
- 出现了新的数据分析和挖掘方法，可以解决之前不能解决的问题。
- 业务的数据意识会随着数据工作的开展而提高，以往的数据知识已经不能满足业务需求。



注意 业务数据意识的不断提高客观上要求数据从业者自身素质和能力也不断提高，通过新技术、新方法、新应用和新发现不断驱动自身成长。这也是数据从业者自身成长路径的重要方面。

举例：以一份专项报告为例，在业务初期建立起专项报告机制。随着业务的发展，数据报告中不断涉及业务新出现规则的数据反馈，同时报告分析思路也在不断调整以适应业务需求；在业务主体的发展到达顶峰时，数据分析思路、维度、方法也达到顶峰。但随着企业内部调整及业务主体的衰落，数据报告的内容也逐渐减少，最终报告随着部门的裁撤而消失。

（5）数据监视

数据监视是对数据的持续跟踪和评估，从技术安全、运维成本和业务应用价值等角度不断优化数据质量，同时为数据存档做准备。

（6）数据存档

数据存档是减少数据运维和存储压力的重要方式，也是提高数据应用效率的主要途径之一。通常，企业内部数据的使用频率随着时间的推移，历史数据的使用频率逐渐下降。而在大多数场景中，对于历史数据的应用都只是汇总统计层面。

数据存档分为两部分，一是在线存档，将访问频率较高且应用性较高的数据从应用中分离出来，通过数据库等形式保存，在此期间用户可以在线访问；二是离线归档，对于访问频率低且应用性较低的数据以光盘等形式进行离线存储，此时用户无法在线访问，只能通过恢复介质访问。

（7）数据销毁

销毁是对已经失去价值或应用性的数据进行销毁的过程，包括删除、擦写甚至物理介质的销毁，这是数据生命周期的结束。该情形常见于业务调整或业务取消等，失去数据所依赖的业务主体。

8.7 本章小结

本章从整体上梳理了企业内大数据工作流程中的主要环节和内容，并按照数据工作进行讲解。其中，非结构化和半结构化的数据计算部分可以看成相对独立的工作流程，原因是不同的工作需求和场景下的流程是不同的。

本章有以下内容需要读者重点掌握：

- 掌握整个大数据工作的基本环节；
- 掌握数据计算部分的全部内容并加以理解和吸收，尤其是深度挖掘部分的内容；
- 重点关注数据质量建设的框架，尤其是数据标准的制定和数据生命周期的管理；
- 熟悉数据应用的场景方式和场景。

企业大数据业务应用

大数据的应用过程需要经过商业理解、数据收集、数据处理、模型计算、模型评估和模型部署这6个阶段，最终实现的目的都是将大数据应用到具体的场景中，这才是大数据真正的初衷。那么大数据都可以应用到哪些场景？这些场景又都是怎么运作的？每个场景实现的过程是怎样的？最后大数据又能帮助这些场景实现或提升什么样的结果？这一切都需要结合具体的业务需求来实践，本章将从整体流程和总览的角度来看一下大数据具体的应用场景组成都是怎样的。

9.1 大数据应用场景概述

要从场景本身出发了解大数据场景应用，可以从两个角度来看，第一个角度是有应用场景的角度，即有什么场景，以及针对这个场景我们要实现什么样的商业目标，再根据场景目标去寻找收集数据内容，最后经过一系列的挖掘过程来实现场景的应用；第二个角度则是没有场景的角度，而只知道我们有什么样的数据，根据数据内容，我们去挖掘对应新的场景应用，实现我们可能想到但无法实现的事情，又或者实现我们都没有想到过的事情。而对没有场景又没有数据的情况，对我们来说目前是没有意义的讨论。因此，大数据的第一个角度是在现有业务的基础上进一步做出的提升，而第二个角度则是通过大数据挖掘创新型业务模型或内容。

我们还可以从大数据的应用对象上来看，有应用于人的大数据，例如利用大数据发现人们之间社交关系的紧密程度，预测异常的违法行为，评定信用等级；有应用于物或事件的大数据，例如利用大数据帮助设备找到更优的运行工况路径，汇集复杂设备的故障成因，帮助发掘异常事件的根源及规律。

从以上的角度似乎不能很好地涵盖大数据的全部应用范畴，不同的视角将会产生不同的

应用点，因此根据已有的经验，我们认为大数据是一门科学，科学知识自然是来自于人类，应用结果也是服务于人类的，因此我们站在用户的角度来考虑大数据的场景应用，其中场景因素包含人、作用对象、发生事件、时间和空间五个方面。为了更好地理解大数据应用场景的实现过程，我们通过电子商务的流程来进行举例。

9.1.1 场景商业目的分析

应用场景的需求分析(如图9-1所示)，分析的角度可以从商业目的出发，从电子商务的目标来说，主要是用户体验的提升、销售的增长、供应链的优化以及内部效率的提升，从场景因素来看，客户以及内部员工是整个流程中人的因素，作用对象则是电商平台销售的产品，发生事件从企业内部来看包含但不仅限于采购事件、营销事件、财务事件、服务事件等，从企业外部来看包含产品查看、产品订购、产品反馈和售后服务等，从时间来看包含客户落地、浏览、注册、订购和反馈时间轴，也包含企业内部的采购时间轴、营销时间轴、财务时间轴和服务时间轴等，从空间来看包含物理空间和虚拟空间，主要包含网站主页空间、频道分类空间、导航空间、主题页空间、产品页空间、比价页面空间、帮助说明空间、客服空间、仓储空间、物流配送空间和送货地址空间等。



图 9-1 应用场景需求分析汇总

9.1.2 场景数据来源分析

企业具有的数据内容分析，数据来源主要包含第一方数据、第二方数据和第三方数据，其中：

第一方数据(如图9-2所示)指企业内部数据，包含用户数据、商品/服务、物流配送、仓储库存、客服数据、人力数据和财务数据等，这些数据中对个性化推荐起决定性作用的主

要是用户数据、商品/服务数据,物流配送、仓储库存和客服数据起辅助作用。一般来说用户数据包含但不限于用户属性、浏览行为、点击行为、注册行为、收藏行为、活动参与、订单行为、积分行为、支付行为、评价行为和售后行为等;商品/服务数据包含商品/服务的分类、品牌、参数、功能说明和生命周期等;物流配送数据包含用户信息、商品信息、配送工具、配送时间、支付信息、配送状态、用户反馈和配送员信息等;仓储库存数据包含商品信息、商品状态、商品调配、库存数量等;客服数据包含咨询数据、建议数据、投诉数据、表扬数据、实时呼叫、客服工作量和质量监控等。

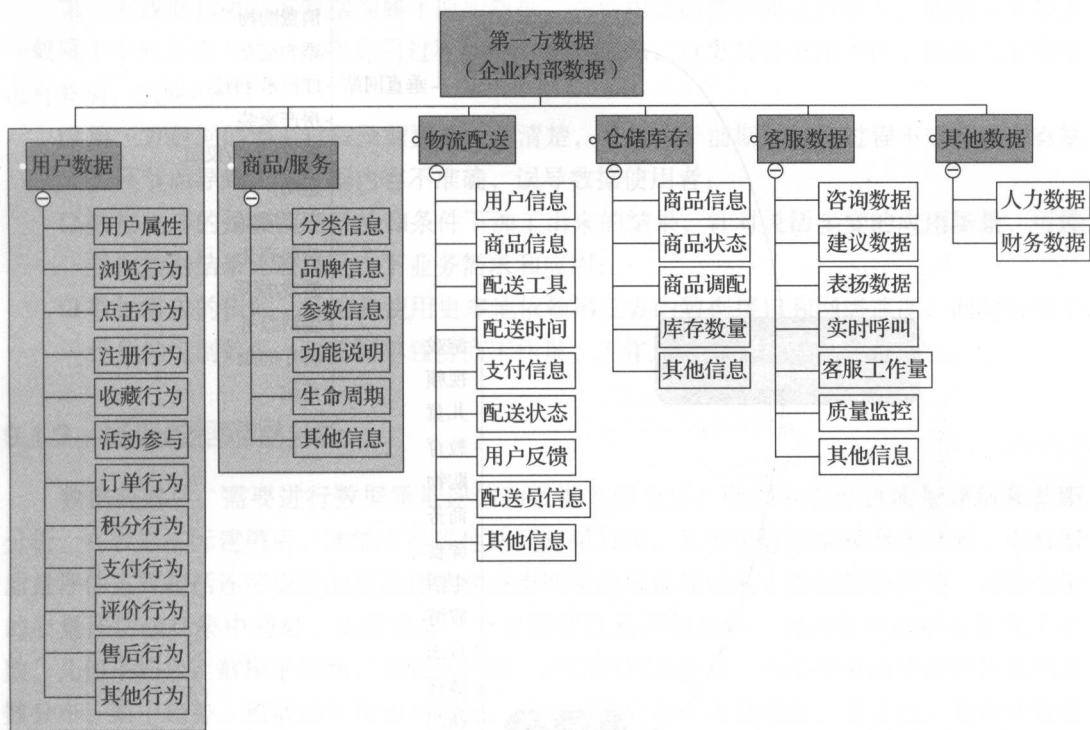


图 9-2 企业第一方数据

第二方数据(如图 9-3 所示)指网络数据,即根据实际业务需求,通过网络爬虫的方式,获取互联网的相关数据信息,例如竞争对手商品/服务分类数据、参数数据、功能说明和价格数据等,该类数据主要用于完善用户行为偏好,自有商品/服务属性,与竞争对手对比优势,制定差异化营销策略,以及个性化推荐的辅助参考等。还有社交网络信息,主要是垂直社区论坛、贴吧、微博等,该类数据主要用于了解用户对企业、品牌和商品等各方面的舆情分析,及时抑制负面信息,同时挖掘用户的社交关系以及对商品或服务的真实需求,以此改善商品功能,完善服务流程,提升用户体验。由于网络信息采集的速度限制、网站自身的反采集措施以及采集硬件的成本问题,在进行第二方数据采集时,需要结合业务实际需求,更具针对性地进行,这样才能更快地实现目标。

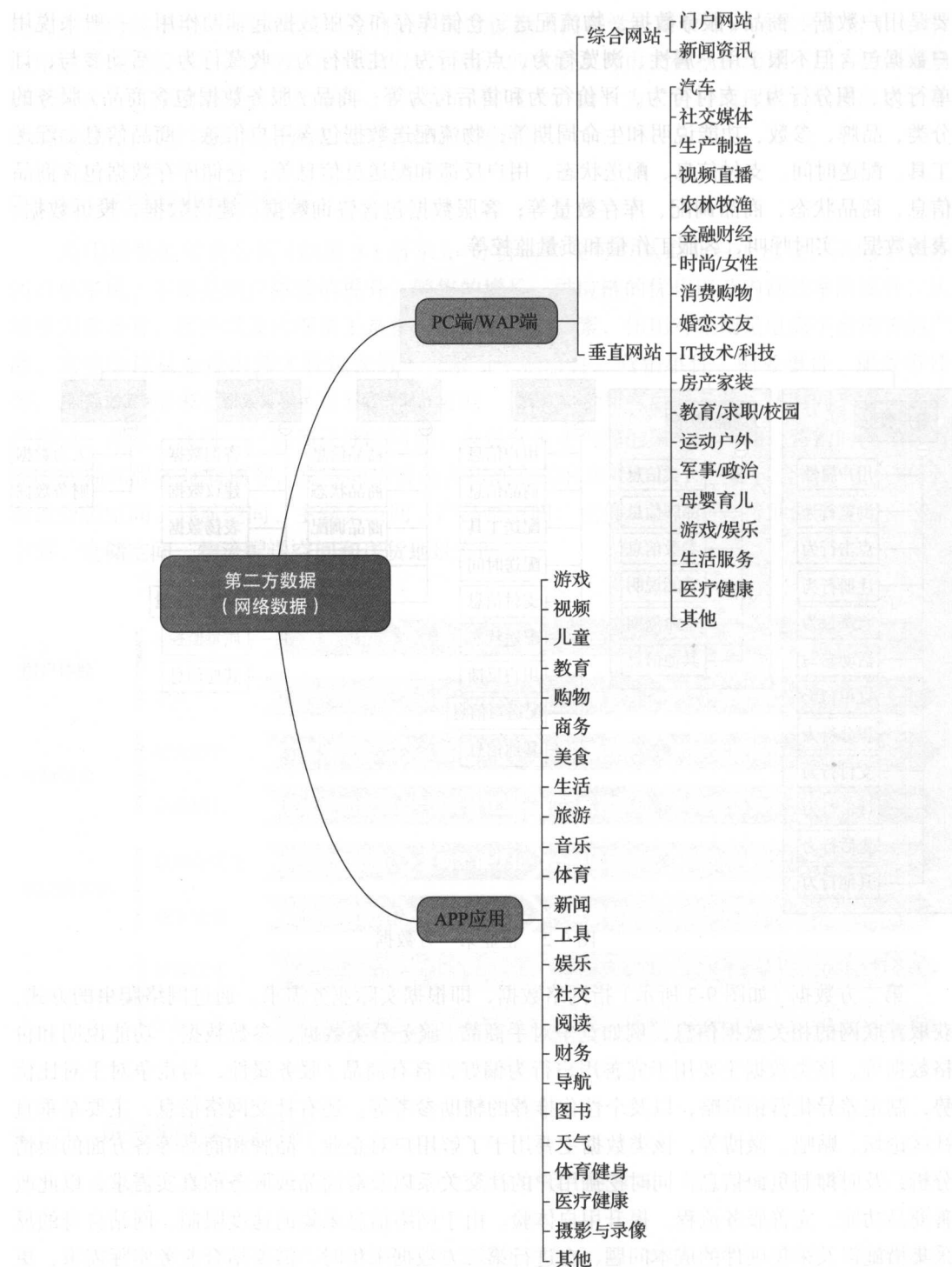


图 9-3 企业第三方数据

第三方数据指合作方或友商提供的外部数据,例如以合理合法的方式通过电信运营商获取处理后的用户标签数据,通过用户的标签数据可以了解用户的基本信息匹配度、用户的网络行为、兴趣偏好、上网习惯偏好和模糊地理信息等,通过这些信息,再结合第一方数据和第二方数据,有策略地进行个性化商品/服务推荐。除此以外,还可以根据用户的综合消费系数、稳定系数、风险系数、终端系数、通信系数、漫游系数、上网系数、交往系数、朋友圈系数、通信监控系数、漫游监控系数,以及对用户的综合信用分数评定等内容,针对用户的信用分数评级,给予用户响应的优惠或特殊权益,增加用户黏性。

第一方数据和第二方数据均属于原始数据,在使用之前都需要进行加工,而第三方数据一般属于半成品或者成品,在应用过程中相对更加成熟,也更具备使用条件。但第三方数据也有弊端,主要包括:

- 第三方数据的加工过程及前置条件不清楚,可能由于前期的清洗过程不规范或者有缺失环节而导致实际数据内容不准确,误导数据使用者;
- 数据更多的是在既定环境和条件下加工出来的结果,针对灵活多变的应用场景,可能导致数据结果无法满足当下业务需求和应用;
- 数据需求的传输、匹配和使用更多地依赖第三方的数据接口和网络速度,同时针对个性化的定制数据,更需要单独协商工作量、工作周期甚至相关费用的产生。

9.1.3 场景数据难易分析

数据获取后,需要进行数据质量评估和数据差距分析,数据只有经过质量评估和差距分析,在数据实际建模后,才能更准确地达到建模目标,从而更好地实现商业目的。而数据质量评估主要包括连续变量的质量评估和分类变量的质量评估两个数据类型评估,连续变量的质量评估包括集中趋势、离散趋势、分布特征以及其他趋势,例如集中趋势有算数平均数、几何平均数、截尾平均数、调和平均数、中位数和众数等;分类变量的质量评估包括频数分布、集中趋势、离散趋势和相对指标,例如频数分布有类别频数、百分比、累计频数和累计百分比等。另外,数据质量评估更重要的一个环节就是查找数据中的空值、异常值,根据业务规则和需求来判定是需要删除或者填充,而填充又需要通过什么方式什么样的值进行填充。

场景数据的差距分析主要是从业务和技术实现的角度来看,根据现有采集的第一方、第二方和第三方的数据与要实现的业务场景,以及需要使用的技术难易程度,通过综合定性和定量分析方法结合的方式来评估实现机会(如图9-4所示)。

9.1.4 场景应用举例

为了确保商业机密,以下场景应用案例仅为了表达部分案例理论、实现过程以及结果展示,真实的数据已经经过处理。

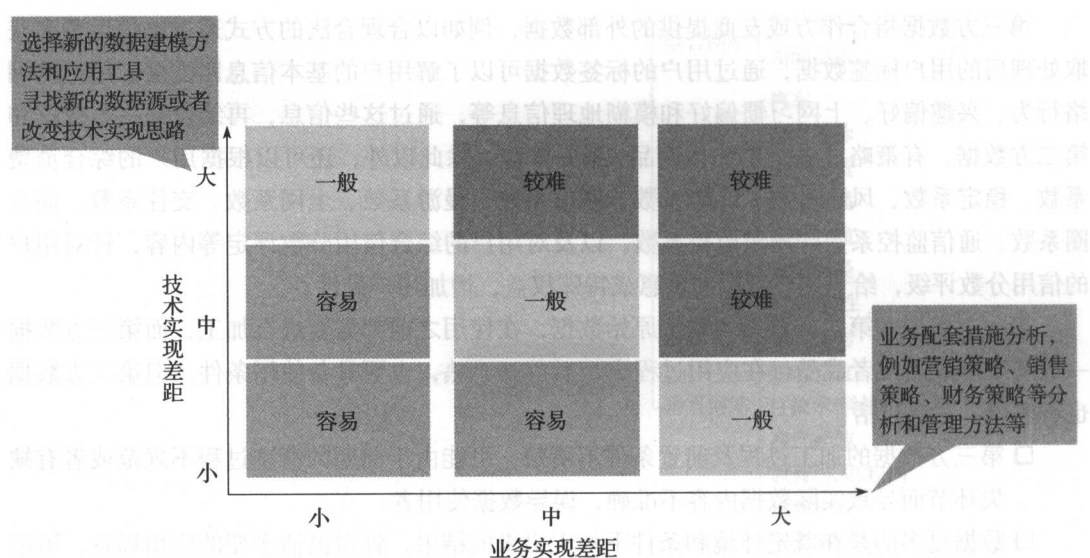


图 9-4 场景实现难易矩阵

1. 互联网零售行业

基于 RFM 的客户关系管理：通过最近一次消费 (R)、消费频率 (F) 和消费金额 (M) 三个因素，并结合其他算法，例如聚类算法，对客户群体进行分群管理和营销的方法（如图 9-5 所示）。

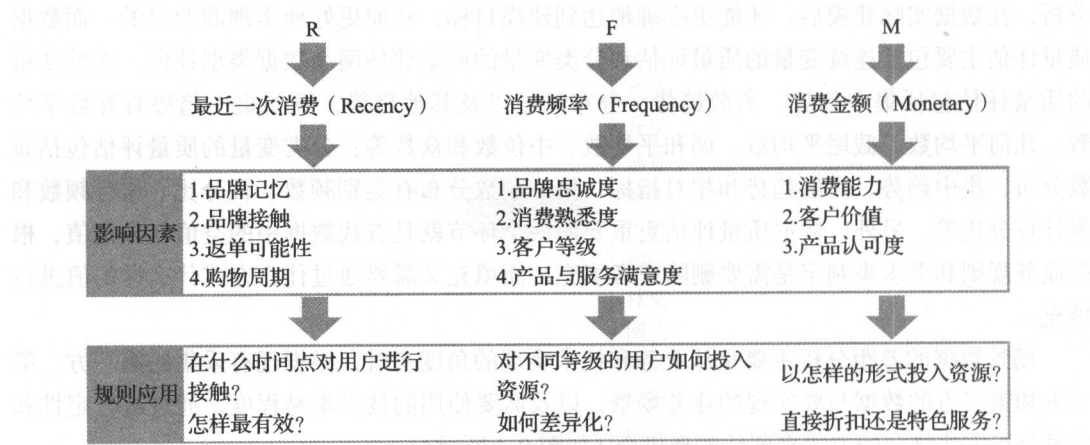


图 9-5 RFM 客户关系管理模型

购物篮分析：根据用户的购买习惯和偏好，向用户展现或推荐相关的商品信息，例如在商品浏览的“猜您喜欢”，购物车中的“买了又买”，收藏夹中的“您可能喜欢”和订单页面的“向您推荐”等，都是通过大数据的关联分析算法完成的推荐功能（如图 9-6 所示）。

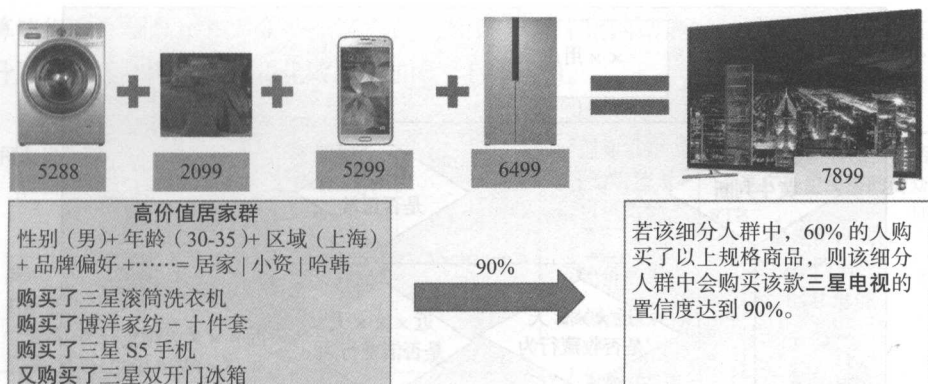


图 9-6 客户购物篮分析

用户行为分析：通过互联网用户在网站中各个页面，甚至各个网站之间的跳转行为，分析用户的性别、偏好和使用习惯，从而进行用户行为分析刻画，运用协同过滤等大数据算法，进行人群分类、营销推荐以及用户体验优化（如图 9-7 所示）。

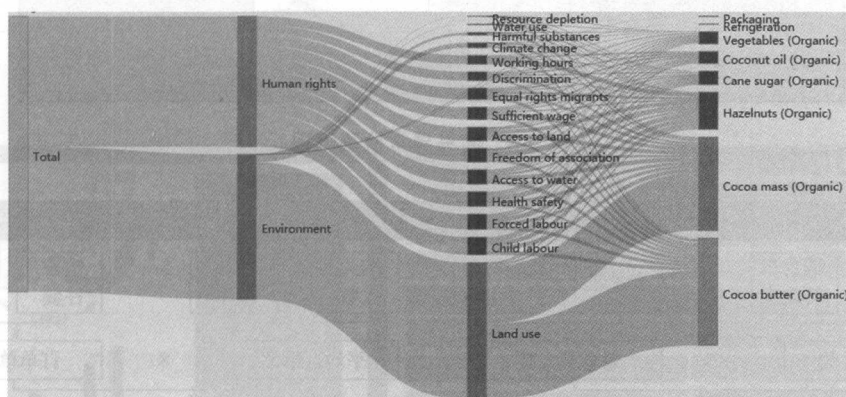


图 9-7 用户行为数据流

异常用户 / 订单检测：使用大数据树类算法，从大量的正常用户和订单信息中，找到黄牛或黑客用户信息，增加企业销售利润，降低企业网络销售安全性、公正性和公平性，提升用户体验，维护企业形象（如图 9-8 所示）。

零售销售成因分析：按照业务规则和意图，将零售销售的成因分为入口成因和出口成因，即根据收入形成的来源与利润流失的渠道来进行分解展示，通过一张图了解企业当期利润的来龙去脉，实现对企业经营的实时把握和管控（如图 9-9 所示）。

网络定向广告：通过用户网络行为、IP 地址、使用客户端进行数据解析，实现用户定向、媒体定向、地域定向、时间定向、设备定向、上下文定向和频次定向等分类信息，实现网络广告的精准投放，从而达到广告投放的千人千面，增加广告展现的相关性，提升用户体验和销售转化（如图 9-10 所示）。

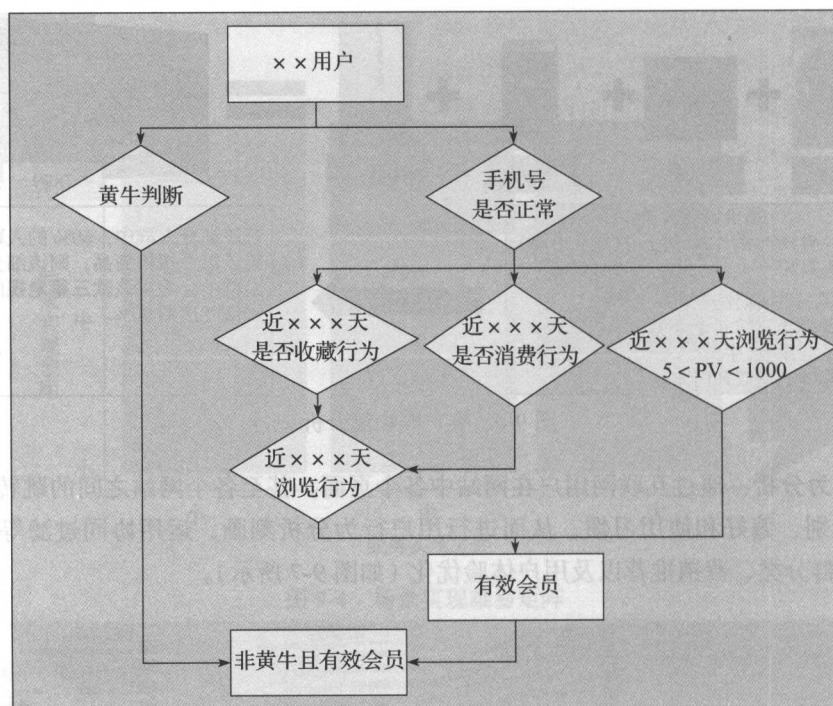


图 9-8 简易异常检测流程

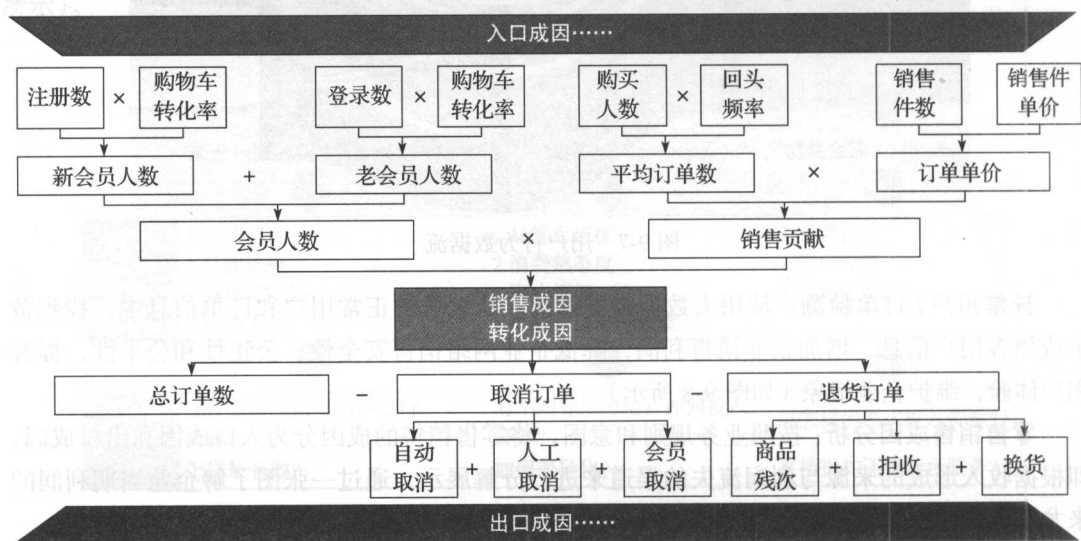


图 9-9 销售成因分析

用户服务感知评价：通过用户对网站的使用，结合关键指标内容，例如用户到达、停留、跳转、退出、加入收藏、加入购物车、订购直至最后成交甚至退换货的过程，使用大数据的

多种算法组合,刻画用户对每一个环节的感知评分,优化网站展示、流程以及运营工作,从而提升用户体验和网站整体转化效果(如图9-11所示)。

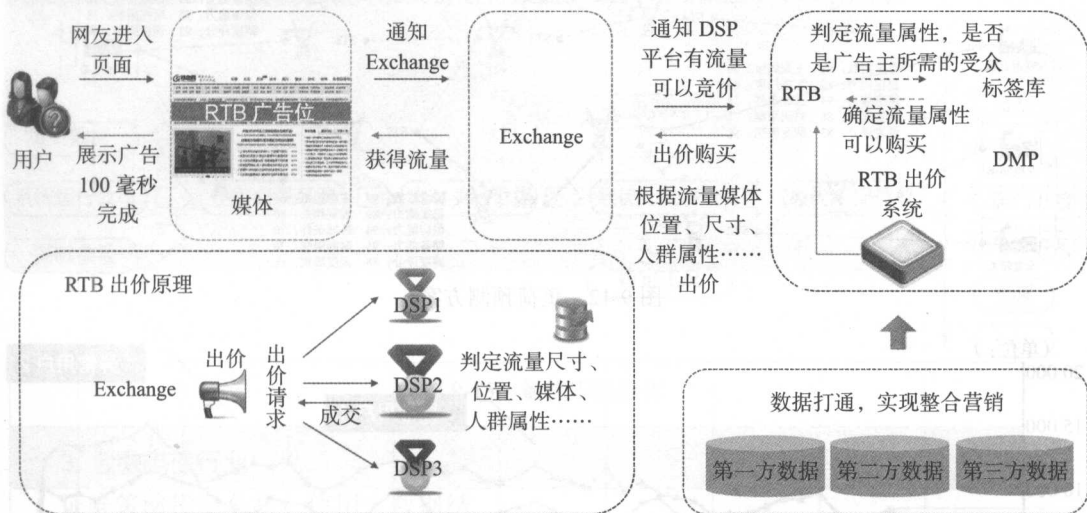


图 9-10 网络定向广告流程

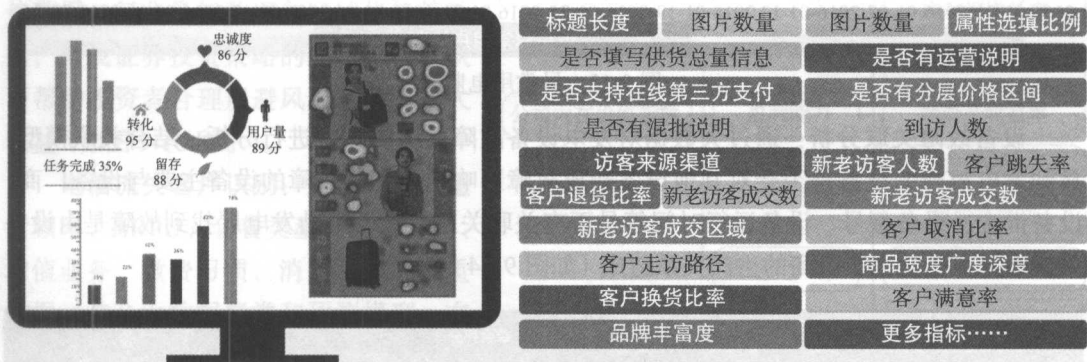


图 9-11 用户评价分析

2. 电力行业

电网负荷预测：使用大数据预测模型，结合发电厂、燃料、气候、地理、距离等综合因素，推算未来电力网络之间的负荷程度，以有效地帮助电网调度对输送电方案进行及时调整，保障电力调度的安全有效(如图9-12所示)。

自适应防窃电实时诊断：通过大数据实时对用电量装置进行检测以及历史数据比对的方式，替代了原来通过人工定期巡检、定期校验电表、用户举报窃电的方式，规避了由于窃电和计量装置故障造成的漏收、少收电费对电力系统造成的损失(如图9-13所示)。

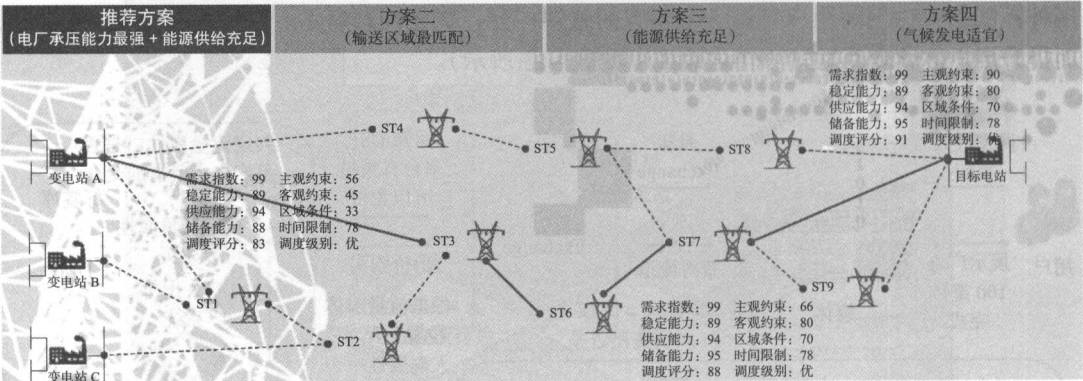


图 9-12 负荷预测方案

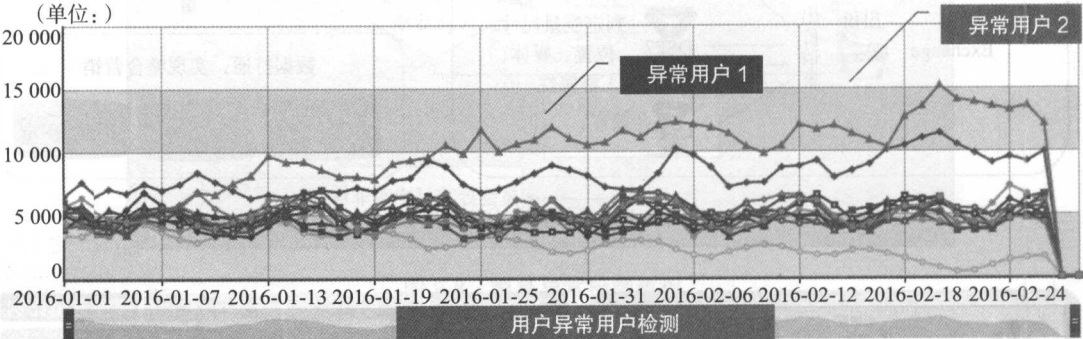


图 9-13 异常用电监测

设备故障关联分析：通过大数据对发电设备故障的历史数据进行分析，结合关联模型，找到发电故障的设备是否会对其他设备造成故障影响，还包括故障的设备主要与设备厂商、设备批次、设备型号、设备运行时间等是否有关联关系，从而帮助发电厂找到故障是由设备单体本身，还是由于厂商的生产关系造成（如图 9-14 所示）。



图 9-14 设备故障关联分析

设备运行最优工况：通过大数据找到设备的最优运行工况，结合设备本身设计值、历史运行数据和当下客观条件数据，拟合成最优工况，通过模拟仿真系统的调试对比，实现对真实设备最优工况的调整，进而实现设备运行效益的最大化（如图 9-15 所示）。

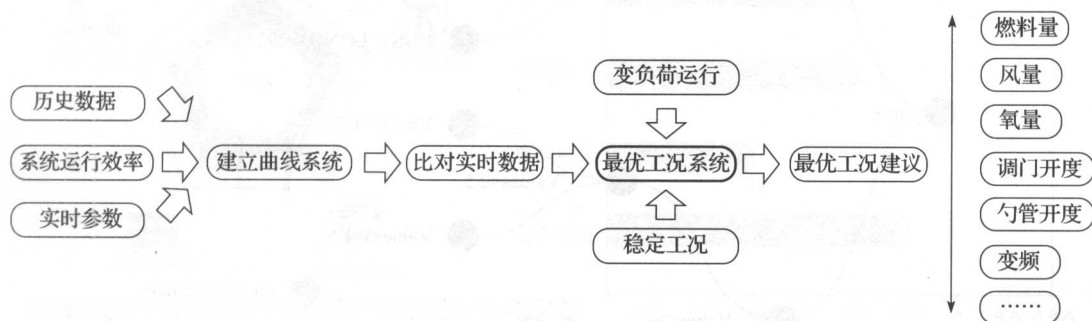


图 9-15 最优工况分析

3. 金融电信行业

证券策略投资分析：使用大数据结合网络爬虫技术，获取需求的业务分析数据，通过企业内在价值、大盘总体趋势数据、行业景气数据和区域优势等数据，寻找证券投资策略的最优分析，从而帮助投资者合理规避风险，获取更大收益（如图 9-16 所示）。

电信流失客户识别：通过客户的通话频率、ARPU、套餐类型、通话时间、增值业务、缴费习惯、消费习惯等多重数据，结合大数据聚类 and 预测模型，在既定业务规则下，找到电信流失客户的共性特征，配合有针对性的营销策略，挽回即将流失的客户，从而延长客户的生命周期，提升客户的 ARPU 值（如图 9-17 所示）。

客户分群智能营销：为了达到更好的营销效果，帮助营销部门合理地管理客户信息，分门别类地传递正确的营销内容，利用大数据结合用户行为、订单和售后信息，利用自学习的算法模型，实现客户的自动化分群管理以及群体信息描述功能（如图 9-18 所示）。

客户征信分析：在信贷数据中，违约客户所占比例较少，一般在 3% ~ 5%，在选取建模数据时，采取分层抽样的方法构成建模样本，根据业务的需要对若干个关键变量通过分层控制，从而使建模数据更具代表性。通过信用评分帮助贷款机构实现贷前审核和贷中跟进的情况，确保将贷款风险降低至最小范围（如图 9-19 所示）。

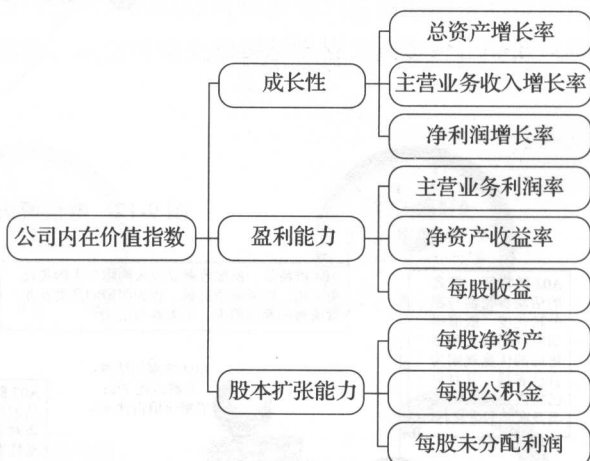


图 9-16 证券策略投资分析

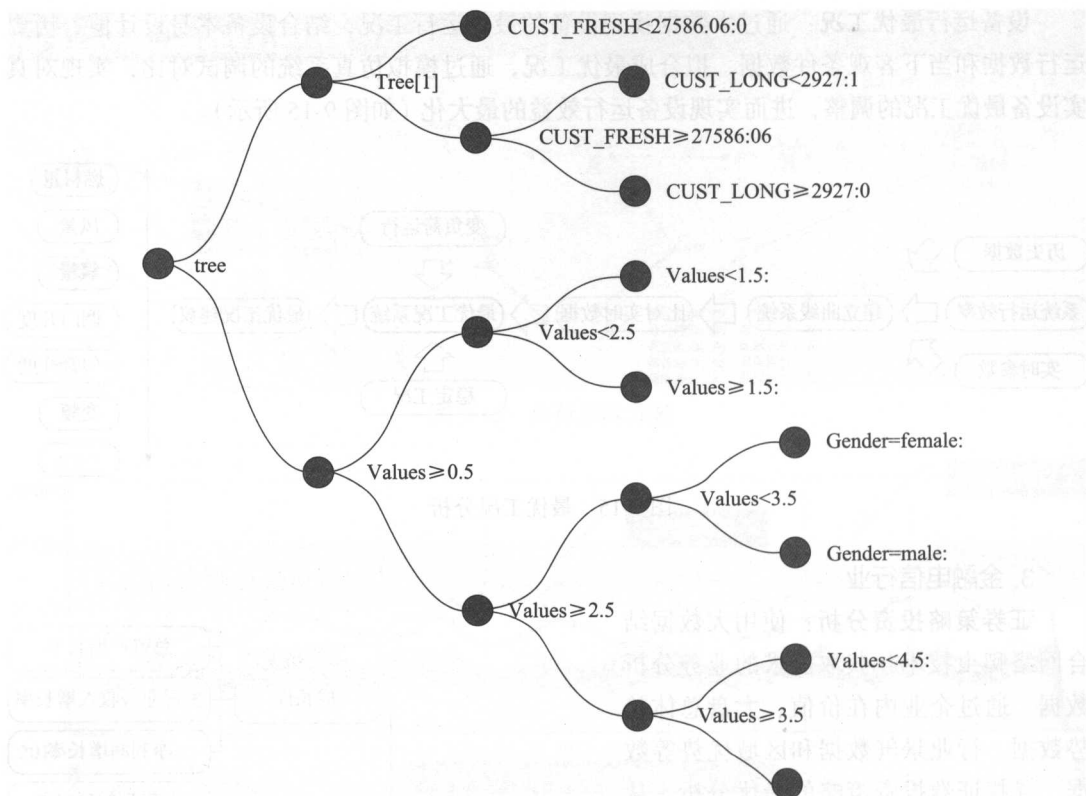


图 9-17 电信流失客户识别

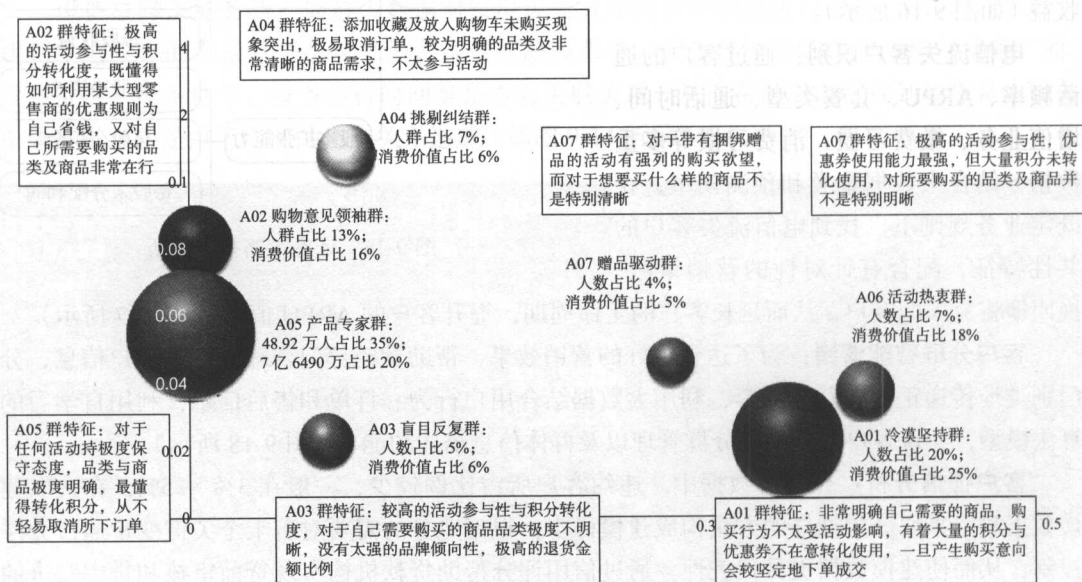


图 9-18 客户分群智能营销

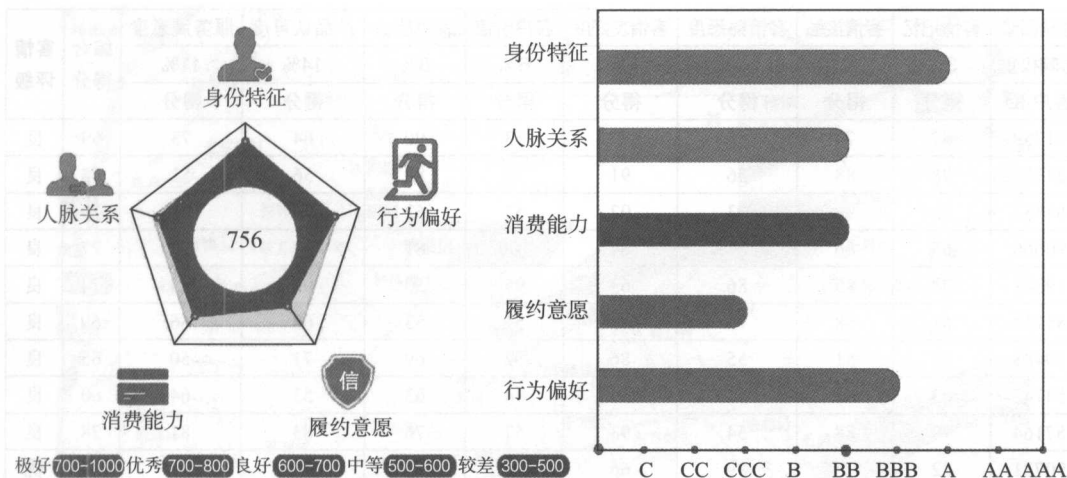


图 9-19 客户征信分析

客户防欺诈分析：通过客户提供的个人信息，与过往的常用信息进行自动化匹配，并根据客户的征信信息，综合判定客户欺诈的可能性，同时还可根据客户提供的消费和报销信息，判定客户骗贷、骗保等金融诈骗的可能性，从而帮助金融保险机构识别客户，降低风险（如图 9-20 所示）。

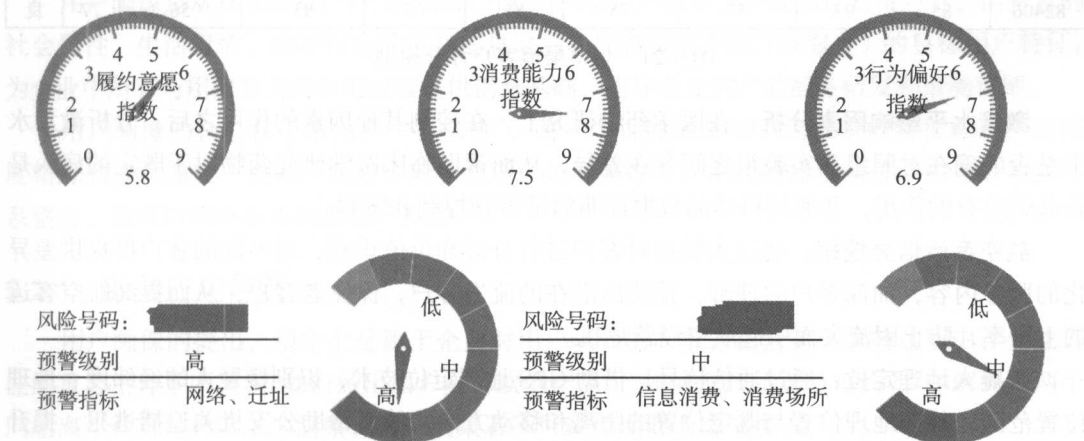


图 9-20 客户防欺诈分析

4. 公共服务行业

烟草渠道客户评分模型：通过零售渠道的商户的客情记忆、客情友好度、服务满意度、客户价值、客情熟悉度、产品认可度、客情接触和客情熟悉度等多个方面，综合评价客户评分，从而帮助渠道维护人员更好地服务客户关系，提升品牌卷烟的销售业绩（如图 9-21 所示）。

影响因素	客情记忆	客情接触	客情熟悉度	客情友好度	客户价值	消费能力	产品认可度	服务满意度	综合得分	客情评级
影响权重	3%	22%	19%	17%	6%	8%	14%	11%		
客户 ID	得分	得分	得分	得分	得分	得分	得分	得分		
91759	67	74	55	77	53	90	64	73	69	良
24359	98	88	56	91	75	80	56	51	73	良
64582	53	98	93	92	53	62	50	67	79	良
90666	65	66	84	64	100	81	87	79	77	良
19341	98	55	86	63	95	50	96	85	74	良
85134	51	58	77	93	51	53	69	66	69	良
39483	77	51	55	86	92	69	71	50	65	良
54742	63	61	57	61	74	63	53	64	60	良
57164	79	88	54	93	57	76	83	81	78	良
90781	82	100	79	66	76	95	76	99	85	优
69861	53	78	75	76	59	76	71	84	75	良
72389	72	68	62	97	75	68	75	64	73	良
93367	79	86	89	96	83	100	58	77	84	优
85322	98	73	81	63	75	53	62	83	72	良
96786	87	59	54	77	97	57	94	73	70	良
69089	96	91	61	94	50	66	81	84	79	良
15747	65	94	52	81	80	97	86	73	79	良
82406	85	97	57	75	85	67	95	56	77	良

图 9-21 烟草渠道客户评分模型

激素水平影响因素分析：在医学药物研究上，在控制其他因素的作用之后，分析激素水平是否的确在对照组和实验组之间存在差异，从而帮助临床医学研究药物对于既定的病人是否起到应有的作用，并通过科学的数据证明病情得到控制和缓解。

航空客运信息挖掘：通过大数据对客户进行分群和价值识别，对不同的客户群提供差异化的服务内容，加深客户的理解，预测出潜在的流失客户，保住老客户，从而提高航空客运的上座率，防止因流失而引起航空经营危机。

嫌疑人地理定位：通过通信信号，借助 GPS 地理定位技术，识别嫌疑人的经纬度、地理位置范围，甚至地理位置与既定位置的距离和移动方向，从而帮助公安机关追捕逃犯，提升破案进度（如图 9-22 所示）。

纳税人偷漏税评估：通过纳税评估指标内容，例如收入指标、利润指标、成本指标、资产指标、负债指标、往来指标和特定指标，并结合第三方的数据内容，通过决策树（例如 C4.5、CART 等），融合偷漏税的群体特征，寻找同样类似的客户信息，规避偷漏税风险。

汽车销量预测分析：利用企业过去一段时期的汽车销售历史数据来预测未来一段时期的汽车销售，结合宏观经济数据、汽车市场行业趋势数据等，使用非线性回归的方法建立曲线预测模型，从而预判汽车销量趋势。



图 9-22 嫌疑人地理定位

9.2 用户画像

用户画像是从真实的用户行为中抽象出来的典型用户模型，企业通过收集与分析消费者的社会属性、生活习惯、消费行为的主要信息之后，完整描述产品（或服务）的目标用户特征，为企业中所有与用户有关的决策过程提供信息基础，指导企业的产品服务研发和市场营销。

用户画像的核心在于给用户“打标签”，每一个标签通常是人为规定的特征标识，用高度精炼的特征描述一类人，例如年龄、性别、兴趣偏好等，不同的标签通过结构化的数据体系整合，就可以组合出不同的用户画像。

9.2.1 业务应用背景

用户画像的提出，根本上是源于企业对用户认知的渴求，在营销决策的过程中，企业关注的重心不外乎两类，“如何做出用户更喜欢的产品”“如何把产品卖给对的人”，解决这两个问题离不开对用户需求的洞察，因此决策者不可避免地要考虑两类人：

- 现有用户：我的现存用户是谁？为什么买我的产品？他们有什么偏好？哪些用户价值最高？
- 潜在客户：我的潜在用户在哪儿？他们喜欢什么？哪些渠道能找到他们？获客成本是多少？

为了回答这些问题，企业必须通过各种方式不断地收集用户信息，最初可能只是通过问卷调查和用户访谈等少量、定性分析的方式进行，当样本的数量逐步提升，这些用户的信息将会以更加标准化、更简单的方式描述出来，形成一个个的“标签”，这也就形成了用户画

像的雏形。因此，用户画像并不是大数据时代的“专利”，大数据技术的应用，拓展了企业获取数据的来源和处理数据的方法，让企业有机会得到更多的用户样本，从海量数据中找到那些真正对自己有价值的数据，从更多维度描述自己的用户画像。

9.2.2 主要实现过程

所谓的用户画像就是将用户进行标签化管理，既然要进行标签化，我们首先需要知道标签化的指导思想，因此我们需要思考以下过程：

画像的对象是谁？

其实标签化的对象就是我们需要关心和希望维护的群体，一般常规的认为，这个群体就是能直接带给企业经济利益的用户，其实这个理解本身没有问题，但其实还可以继续扩展为帮助企业带来业绩的员工，还可以是企业上下游的合作伙伴等，因为企业的整个利益生态是和他们密不可分的，因此用户画像可以是一个广义的概念，根据企业期望实现并能够带来响应业务目标的对象，都可以是用户画像的对象（如图 9-23 所示）。

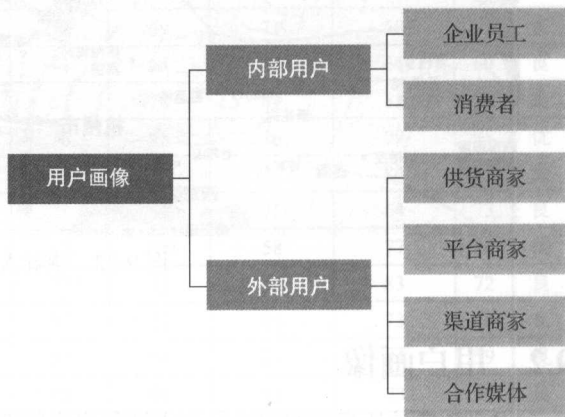


图 9-23 用户画像对象

用户画像对象不仅限于上述描述范畴，根据企业自身特性，还可以进行更多的扩展，但目的都是需要能够给企业带来实际的商业价值。

画像的目的和内容是什么？

画像根据不同的对象，其应用价值也是不一样的，各有各的目的和意义，但对于企业来说都具有共同的特点，即通过用户画像帮助企业实现商业价值的提升，我们单纯以电子商务行业来进行画像目的的说明，仅作为参考和借鉴。

企业员工画像：通过员工身份、属性、兴趣、历史工作、家庭和目前工作现状等，可以形成完整可用的有关员工的工作状态，方便对员工的完整认知。通过这些画像分析，可以用于员工需求分析、技能培训分析、素质分析、人才储备分析以及员工在公司的生命周期管理，从面试、入职、培训、工作、晋升、薪酬、异动、人际关系、团队合作、业绩贡献直至流失的全过程，帮助人力资源提升各方面的管理，例如简历匹配、应聘人员量化比对、猎头合作、岗位素质要求、绩效考评、关键员工培养、员工关系管理和团队协作等。

消费者画像：使用用户自愿填写的信息、网络浏览轨迹、消费购物信息、售后服务信息等，形成用户的前景式画像，方便把握用户的需求和习惯。画像结果可用于用户需求判别、消费习惯分析、网站用户体验分析、个性化营销、精准广告、消费者访谈以及消费者全生命周期的管理，帮助企业了解消费者、优化网站页面布局、提升服务质量、降低营销成本、布

局商品结构和优化服务体系等。

供货商家画像：我们可以使用企业的采购订单信息、商品销售信息、供货及时性信息、供货商家的企业资质、工商信息、关联企业信息等刻画供货商的画像内容，以便于了解供货商的实力、服务及时性和合作配合度。画像结果可用于对供货商进行综合评估，并进行分类管理，不同供货商采取不同的合作策略，且这些策略都是有数据支撑的，不是单纯凭借采购人员的一些话术来决定合作策略、采购策略和财务策略。

平台商家画像：平台商家即指通俗的网站店铺商家，是在网站平台上进行商品销售的商家。通过商家的资质、工商信息、商品类别、网站流量、商家促销、客户评价、商家客服、订单信息和售后信息进行平台商家画像，可用于对商家进行综合评定，帮助商家分析业绩提升办法，例如流量大转化少的原因、没有流量的原因、商家页面设计合理性、活动促销的优劣势、消费者对商家的信赖程度，甚至对平台商家的信用等级进行评分。

渠道商家画像：指帮助电商企业进行大客户或特殊渠道销售的商家，与平台商家的数据画像内容类似，但作用不同，通过渠道商家的画像，可用于对渠道商家的综合贡献能力、忠诚度等进行评定，例如贡献能力包含销售贡献能力、利润贡献能力、口碑贡献能力、媒体支持能力等多个方面。

合作媒体画像：企业要实现销售，与媒体合作是其中一个最有效的方式，但是并不是每个媒体都能够很好地帮助企业实现应有的回报，因此我们要通过与媒体合作的效果、媒体影响力等方面对其进行画像，例如媒体的影响力包含媒体的覆盖地域、行业领域、影响群体梳理、影响群体特征、群体的活跃程度、群体对企业信息的响应程度和贡献能力等。通过合作媒体画像，能够帮助企业降低营销成本，选择更加合理的合作媒体，更好地提升企业销售业绩。

画像后存放位置在哪？

画像的目的是应用，我们所谓的存放位置并非指物理的位置，而是应用的位置，有几种方式可以考虑，一种是以应用场景为驱动，这种方式我们应该把画像结果存放在对应的应用系统软件中，供业务人员直接调用和使用，例如员工画像应该存放在人力资源管理系统中，消费者画像存放在CRM（客户关系管理系统）或者市场营销系统中，供货商家画像存放在财务系统或者采购管理系统中，平台商家存放在网站运营管理平台中，合作媒体画像存放在营销管理系统中。另一种是以数据为驱动，这种方式的前提是有统一的大数据一体化平台的支撑，我们将画像内容统一放置在该平台上，供业务人员调用。

上述两种方式各有利弊，分散存放是在不改变目前业务流程的情况下，直接优化目前业务系统的功能，好处是不影响使用人员系统，能够快速投入应用，弊端是信息数据以及画像功能的维护分散，不利于统一的管理，更新起来较慢。统一存放则与分散存放相反，使用上需要跨平台，但整体数据的管理和优化更加快速和方便，解决上述两种弊端的办法就是建立一体化的大数据平台，将目前的应用功能集成到一体化平台上来，但这种办法就是前期投入时间较长，需要一定的资金支持。当然除了这些，相信还有更好的解决办法，读者可以集思广益，融合不同存放方案的优劣势。

不论如何存放和应用，在所有平台的使用上，建议搭配帮助或画像标签的含义解释，因为随着标签内容的增加，开发者或者标注标签的人员与使用标签的人员理解不同，可能会造成标签的错误使用。

画像的更新频率是怎么样的？

画像的更新频率根据业务需求和应用场景的不同各自单独定义，例如员工的画像部分，对于静态信息可以选择以周或者月度进行更新即可，而对于动态信息最好是以小时（或分钟）、日或者周进行更新。静态信息指员工身份、家庭、学历、培训、薪酬绩效等内容；动态信息指工作内容、操作系统行为、上传下达等。为什么动态信息有的需要以小时（或分钟）进行更新，尤其对于电商企业。若员工在操作系统修改关键信息时，可能有操作失误的情况，例如将价格修改错误、将优惠政策或促销政策填报错误的情况，在画像标签更新时，系统需要有及时的判别能力，并做出错误预警，帮助企业挽回损失，同时也能帮助员工减少错误操作。

再如，消费者画像部分也是如此，对于用户体验提升，可以天、周进行更新即可，但对于精准广告，则需要以秒、分钟的级别进行更新，因为用户有可能在看完网站后，直接跳转到外部网站进行其他信息的浏览和操作，例如用户刚刚看完网站上的一款手机产品，此时正在犹豫阶段，暂时跳转到新浪网站查看新闻，而在新浪网站的广告栏位即使提供了该款手机以及相关的其他品牌或价位的手机，用户也可能会重新再回到网站继续对比和查看手机信息，从而提高网站对老客户的召回，提升销售转化率。

因此，画像标签的更新频率需要根据业务要求，有策略、有选择性地频率调整，因为更新频率过快，会大量地占用 IT 资源，影响其他业务的正常进行；更新频率过慢，会影响业务本身的进程，甚至失去用户画像本身的意义。所以画像的更新频率需要结合业务规则和业务应用场景来进行设定。

我们在此仅以消费者作为画像对象示例（如图 9-24 所示），阐述画像的主要实现过程，其他类型的用户画像道理是类同的。一个画像标签通常是根业务规则需要高度提炼的用户特征标识，例如性别标签：男，年龄段标签：35 ~ 45 岁，地域标签：广东。用户画像标签呈现出两个特征：分别是简洁短语和内容语义化，简洁短语是指每个标签通常只表示一种含义，没有过多的其他含义，而且标签本身不需要再做过多文本分析等预处理工作，这也为大数据平台系统提取标准化信息提供了便利。而内容语义化是指使用者能够很方便地理解每个标签的含义，这也使得用户画像模型具备实际业务意义，也便于满足业务应用的需求。

技术或业务人员通过制定标签规则，实现用户标签化，并能够通过快速读出标签信息，大数据系统也方便做标签抽取和分析。所以，用户画像，即标签，向我们展示了一种简单直接的方式来描述用户信息。

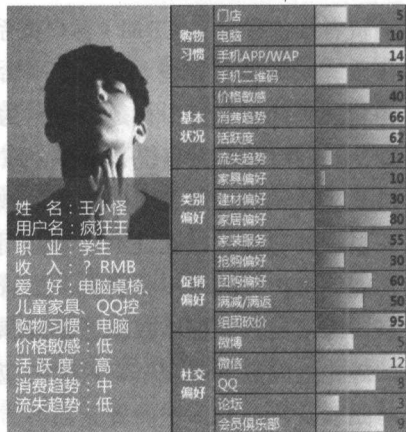


图 9-24 用户画像示例

1. 数据源收集分析

构建用户画像是为了刻画用户的属性、偏好和习惯信息，从而还原用户的原本意图，因为画像数据来源于用户相关的属性和行为数据。

为了更完整地对用户进行画像，我们需要在闭环的思路指导下进行数据画像，这样不用担心架构上对每一层分类没有考虑全面，由此造成维度遗漏引起扩展性的隐患。所谓的闭环思路如用户生命周期分为潜在用户、用户获取、上升用户、成熟用户和流失用户；从用户价值划分看，可以分为三类：高价值用户、中价值用户、低价值用户。所有的子分类构成了分类的总集合，这样的分类方式，有助于后续不断枚举和迭代补充遗漏的信息维度。另外，在本节开头也提到，不同的分类方式适用不同的应用场景，因此标签的分类也应该根据不同的业务需求，按需划分即可。

结合闭环思维方式，我们将用户的相关数据划分为静态数据和动态数据两大类。

静态数据：用户相对稳定的信息，主要指人口基本属性方面的数据。这类信息无需过多地建模预测，大部分直接可以当成标签使用，例如性别、年龄、职业、职务、收入、学历和地域等。

动态数据：用户不断进行的行为数据，例如用户打开网页，查看微博，浏览微信朋友圈，在电商网站上买了一件衣服等一样都是用户行为。当用户的行为集中到电商上时，用户行为就会聚焦很多，例如用户通过导航网站来到该电商网站，浏览商品页面，注册登录，收藏商品，参与抽奖、签到活动，在线客服咨询，将商品加入购物车，提交订单，进行支付，查看配送信息，收货确认和评价晒单等都是互联网用户行为。在此以电商用户为主要分析示例，暂不考虑合作数据和线下数据。用户在网站上的网页浏览行为可以看做动态数据的唯一来源。

2. 画像标签实现目标

用户画像的目标是通过分析用户行为，为每个用户打上标签，以及该标签的权重。标签刻画了用户对页面内容有兴趣、商品偏好和内在需求等。权重即标签的重要程度指数，用户的兴趣、偏好指数，也可能表示用户的潜在需求，可以简单地理解为喜好程度和关注程度。

3. 数据建模方法

如何根据用户行为，构建模型形成标签与权重。一个事件模型包括：时间、空间（地点）、人物、对象和事件行为五个要素。每一次用户行为本质上是一次事件发生，可以详细描述为：什么用户，在什么时间，什么地点，对什么对象，做了什么事。

什么用户：即用户类型，一般是通过特定标识对用户类型进行区分，在零售电商消费者领域可以划分为零售用户、大用户、黄牛用户、恶意用户和企业用户等。另外，有些企业对外也会根据会员等级进行分类，例如钻石用户、金牌用户、银牌用户、铜牌用户和注册用户，而对内除了高中低价值分类，还可能根据用户生命周期所处的阶段进行划分。

什么时间：时间包括两类信息，时间戳+时间长度。时间戳，为了标识用户行为的时间点。比如，2016年10月2日13:50:12（精度到秒），通常采用精度到秒的时间戳即可，因为微秒和毫秒的时间戳对于用户行为标签的影响度不大，除非用来识别虚拟用户（也叫机器人用户，即通过快速注册的方式来攻击电商网站的注册机器人）。浏览器时间精度，一般准

精度最多到毫秒或者秒级即可。另外，时间戳也可识别出其他信息，该类信息有季节信息、早中晚信息、节假日信息，例如 2016 年 10 月 2 日为秋季，13:50:12 为中午，同时也是国庆节期间；同时通过外部数据结合，还可以获取气候信息等，从而构建一个以时间为维度的场景，例如 2016 年 10 月 2 日，中午，季节秋季，国庆节，天气小雨，适合向用户推荐秋冬装、夹克或者棉衣，同时对于有快递优势的电商，还可即时推荐雨具，方便外出游玩的用户出行。另一类时间是时间长度，是为了标识用户在某一页面的停留时间，合理的停留时间也可以标签化为用户的忠诚度，或者对商品的意向深度等。

什么地点：即用户在电商网站的接触点。对于每个用户接触点，潜在包含了两层信息：网址+内容。网址：每一个 URL 链接（网页链接地址，例如 www.baidu.com），URL 定位了一个电商网站的登录地址，或者某个产品的特定介绍页面。可以是 PC 上某电商网站的页面 URL，也可以是手机 APP 或者 WAP 页面上的微博或微信等应用的某个功能页面，某款产品应用的特定画面。比如，手机单品介绍页、微信订阅号页面、游戏的过关页等。内容：每个 URL 网址中的内容。可以是商品的介绍信息：商品分类、商品品牌、商品描述、商品属性、商品功能、帮助信息和网站客服等。比如，苹果、手机，对于每个互联网接触点，其中网址和时间决定了标签权重；页面信息决定了标签内容。

接触点可以是网址，也可以是某个产品的特定功能界面。比如，同样一瓶矿泉水，超市卖 1 元，火车上卖 3 元，景区卖 5 元。商品的售卖价值，不在于成本，更在于售卖地点。标签均是矿泉水，但接触点的不同体现出了权重差异。这里的权重可以理解为对于矿泉水的需求程度不同，即愿意支付的价值不同（如表 9-1 所示）。

表 9-1 接触点示例

标签	权重	接触点	标签
矿泉水	1	超市	矿泉水
矿泉水	3	火车站	矿泉水
矿泉水	5	景区	矿泉水

类似的，用户在国美电商网站上浏览红酒信息，与在酒仙网上浏览红酒信息，表现出的对红酒的喜好度也是有差异的。这里的关注点是不同的网址，存在权重差异，权重模型的构建需要根据各自的业务需求进行，同时在构建过程中也要考虑停留时间以及时间长度的因素。

什么对象：对象即用户行为的作用对象，在电商网站中，主要指虚拟的网页内容，但从实际需求出发，对象分为实物对象、虚拟对象和服务对象。实物对象指具有实体的商品，例如电视、红酒、手机等；虚拟对象指手机话费、游戏币等虚拟商品；服务对象指网站提供的服务内容或售后增值内容，例如延长保修服务、售后上门服务等。

做了什么事：用户行为类型，对于电商有如下典型行为：浏览、添加购物车、搜索、评论、购买、点击赞、收藏等。

不同的行为类型，对于接触点的内容产生的标签信息，具有不同的权重。比如，购买权重计为 5，收藏权重计为 3，浏览计为 1（如表 9-2 所示）。

表 9-2 事件权重示例

标签	权重	行为类型
手机	1	浏览
手机	3	收藏
手机	5	购买

综合上述分析,用户画像的数据模型,可以概括为下面的公式:用户标识+时间+行为类型+接触点(网址+内容),某用户因为在什么时间、地点、针对什么对象、做了什么事,所以会打上××标签。用户标签的权重可能随时间的增加而衰减,因此定义时间为衰减因子 r ,行为类型、网址决定了权重,内容决定了标签,进一步转换为公式:

标签权重 = 衰减因子 × 行为权重 × 网址子权重

比如:用户A,昨天在酒仙网浏览一瓶价值238元的长城干红葡萄酒信息。

□ 标签:红酒、长城、干红。

□ 时间:因为是昨天的行为,假设衰减因子为: $r=0.8$ (需要根据实际业务规则或算法进行调试和优化确定)。

□ 行为类型:浏览行为记为权重1。

□ 地点:酒仙网红酒单品页的网址子权重记为0.8(相比国美红酒单品页的0.6)。假设用户对红酒出于真的喜欢,才会去专业的红酒网选购,而不再综合商城选购。

则用户偏好标签是:红酒,权重是 $0.8 \times 0.6 \times 1 = 0.48$,即用户A:红酒0.48、长城0.48。

上述模型权重值的选取只是举例参考,具体的权重值需要根据业务需求二次建模,这里强调的是如何从整体思考,去构建用户画像模型,进而能够逐步细化模型。

下面以简单的宏观用户画像为例(如图9-25所示),展示如何在大数据分析平台上实现用户画像应用:

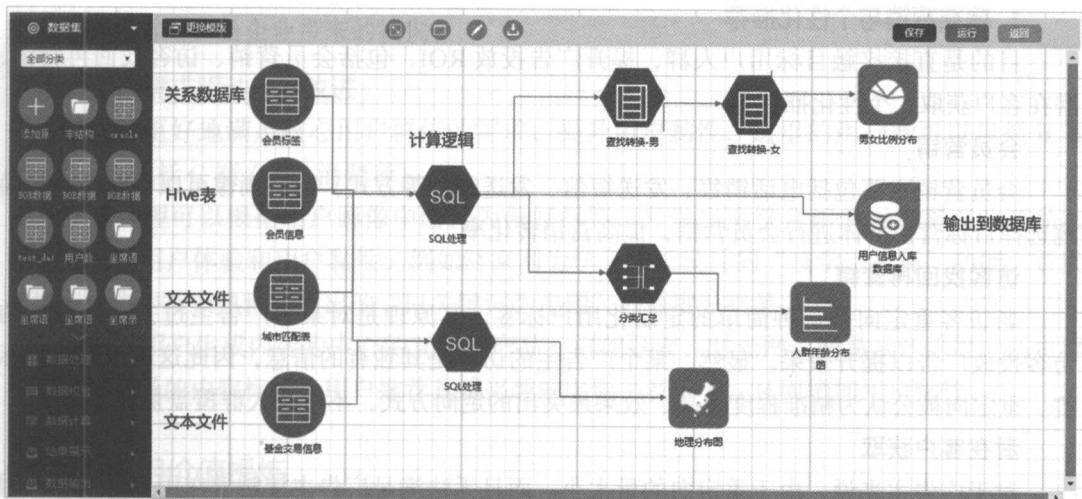


图 9-25 用户画像计算示意

- ❑ 左侧 4 个表为企业通过业务系统收集的原始用户数据；
- ❑ 中间的两个 SQL 处理为计算逻辑，标准计算过程应该是将会员数据标签化；
- ❑ 后面跟着的是数据转换过程；
- ❑ 最后是数据可视化过程，将汇总后的数据表格展示为各类图表。

输出效果如图 9-26 所示。

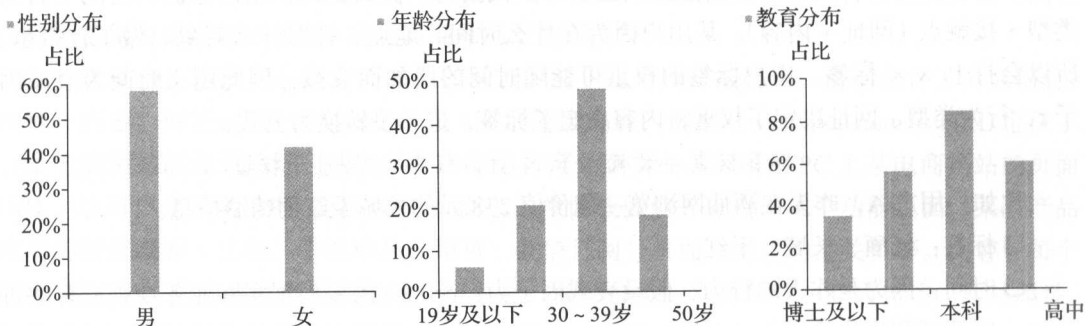


图 9-26 用户画像输出结果

9.2.3 关键应用场景

零售电商企业关键应用场景可大致分为以下 3 类（如图 9-27 所示）。

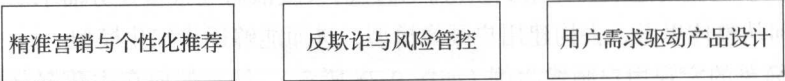


图 9-27 应用场景分类

1. 精准营销与个性化推荐

目的是直接接触目标用户人群，提升广告投放 ROI，包括会员营销、访客找回再营销、潜在客户获取、个性化推荐等。

会员营销

会员营销主要包括电话销售、发送短信、发送邮件等方式，用户画像可以帮助企业精确地选择目标营销群体进行会员营销，提高营销转化率。

访客找回再营销

老访客重定向是一种简单的定制化用户标签，其原理是对某个广告主过去一段时间的访客投放广告以提升效果。显然，某个广告主的访客是其独有的信息，因此这属于定制化标签。重定向被公认为精准程度最高、效果最突出的定向方式，不过其人群覆盖量往往较小。

潜在客户获取

对于广告主来说，由于重定向的量太小，而且无法满足广告主接触潜在用户的需求，因此不能仅仅依靠它来投送广告。潜在客户获取的思路是根据广告主提供的种子访客信息，结

合广告平台更丰富的数据,为广告主找到行为上相似的潜在客户。这一方法的目的是希望在同等用户覆盖比率的情况下,达到比一些通用的兴趣标签更好的效果,这也从实质上体现了广告主数据的核心价值。

个性化推荐

对于电子商务网站,随着销售规模的不断扩大,商品的个数和种类快速增长,如果没有个性化推荐系统的应用,消费者将会花费大量的时间才能找到自己想买的东西。而用户画像的应用,一方面可以帮助电子商务企业分析用户群体的兴趣偏好,从而投其所好地进行个性化商品推荐;另一方面可以帮助电子商务企业找到行为、兴趣相似的用户,从而进行交叉式的个性化推荐。

对于以用户生成内容为中心的网站,例如社交网络、电影评价网站、音乐评价网站、新闻推送网站等,用户画像可以帮助这些网站了解每个用户的兴趣爱好并进行个性化推荐,从而显著提高用户的留存率、使用时长、使用黏度。

2. 反欺诈与风险管控

目的是通过黑名单管理的方式减少异常用户给企业带来的损失,包括黑名单用户画像、用户异常行为检测等。

黑名单用户画像

无论是电信欺诈,还是金融欺诈等,最核心的都是需要企业从用户数据中找到行为异常或者历史记录不佳的用户,并给这些用户贴上相应标签,从而勾勒出“异常用户画像”,然后通过黑名单等方式来规避风险,大幅降低这些用户给企业带来的损失。

用户异常行为预测

将机器学习算法应用于企业历史上的异常用户画像标签数据,从而对现有用户进行预测,找出可能存在风险的用户,从而在电信欺诈、银行客户信贷违约等事件爆发前采取措施,降低异常用户给企业带来的损失。

3. 用户需求驱动产品改进

目的是更好地满足核心用户群体的需求,提升用户黏性,这里的用户既可以包括B端(企业),也包括C端(个人消费者)。

产品经理可以根据用户画像中的行为偏好标签,开发出更符合相关用户需求的产品功能,以提升用户体验和用户黏性,例如链家网为学子家长设计的“按学区找房”功能,百度根据用户集中搜索热词开发出的“以关键词为主题”的百度贴吧等。

对于以用户生成内容为中心的互联网产品,例如豆瓣、时光网、网易云音乐等,还可以按照用户喜爱的内容标签为用户推荐其感兴趣的电影、音乐等,从而增加用户黏性。

9.2.4 应用价值提炼

用户画像作为企业目标用户的真实写照,在企业不同决策环节中都将发挥作用,归纳

起来主要是以下三个方面。

1. 对内指导完善产品设计

在设计互联网产品时，我们很容易落入这样的陷阱，即认为我们正在为理想化的用户设计产品，而理想化的用户就是“某些和我们完全一样的人”，但事实上我们并不是为自己设计；而是为其他人设计，如果想要这些人喜欢并使用我们的产品，我们就必须了解“他们是谁”以及“他们的需求是什么”。

确认用户需求是复杂的，因为用户群体之间存在很大的差异性，所以我们需要对用户进行细分，通过将用户分成更小的群组，每一群用户都具有某些共同的关键特征，这时我们可以通过创建用户群画像的方式，来确认特定每个用户群体的关键特征。从而让我们在产品设计的过程中抛开个人喜好，将焦点放在目标用户的动机和行为上进行产品设计。

2. 对外推动精准营销

十多年的实践证明，互联网最有效的商业模式莫过于可以把流量直接变现为在线广告模式。从最初铺天盖地的横幅广告起步，到人群及兴趣精准定向的搜索广告与推荐引擎，直到与内容环境融为一体的原生广告，用户需求与口味的不断变迁促使着广告产品与技术持续不断地升级与发酵。

互联网广告是一个千亿级的市场，如果把互联网比作一辆车，互联网广告就是汽油，目前中国大多数网站以及众多的移动端 App 均依靠广告盈利。而大多数广告产品的基础是按照受众售卖，因此受众定向是其非常重要的支持技术。

从人口属性、网站频道、网站上下文、产品使用行为、地理位置、访问记录等方面对用户进行勾勒，为每一个用户创建一个“标签”化的用户画像，从而帮助广告主精准地把他们想要传达的信息传达给目标用户群体，实现广告主、媒体、广告平台、用户的多方共赢。

3. 行业用户研究

为深入了解行业发展的动态，我们可以进行用户画像的分析，例如 90 后人群的消费偏好趋势分析、高端用户青睐品牌分析、不同地域品类消费差异分析等。这些行业的洞察可以指导平台更好的运营、把握大方向，也能给相关公司（中小企业、店铺、媒体等）提供细分领域的深入洞察。

用户画像的适用行业与应用如下：

- ❑ 电信公司——会员营销、精准营销、反欺诈分析、客户黑名单。
- ❑ 银行、券商、保险、互联网金融公司——精准营销、贷款风险管控。
- ❑ 电商、零售企业、经销商——精准营销，个性化推荐。
- ❑ 在线旅游、在线教育——用户兴趣、精准营销。
- ❑ 线上社交平台——用户行为偏好、用户兴趣、用户心理、用户关系网络。
- ❑ 线上评论平台（影评、乐评、动漫等）——用户内容偏好、个性化推荐。
- ❑ 大型网游——用户价值等级、用户忠诚度、用户心理。

9.2.5 场景总结回顾

用户画像渗透到互联网行业的方方面面，在电信、广告、传媒、电商、社交、出行旅游、金融、娱乐、教育、O2O 等方面都有用武之地，其创造的价值主要在于帮助企业更好地了解每个客户的特点，然后有针对性地进行营销活动和内容推荐，从而让用户感觉到自己用的产品非常懂自己，这样用户就会花更长的时间用这个产品，或者是在里面消费，从而使互联网企业和用户达成双赢。

- 社交用户画像（Facebook、人人、QQ 空间等）。有很多社交的注册用户，为了增加用户之间的社交文化，对每一个用户进行画像，根据画像做好友推荐。指标：所在地；故乡；性别；年龄；在线；附近距离。
- 网站用户行为画像（各大型网站）。根据网站用户行为画像，对网站性能负载进行综合调整、评估、优化。指标：PV、UV、IP、PR、响应时间、停留时间、跳出率、回访率等。
- 用户群体画像（通用应用）。对不同职业、不同技术背景的用户群体做用户画像分析。指标：所在地；年龄；职业；消费能力；目标客户细分。
- 精准广告营销（腾讯广点通、阿里妈妈、谷歌 AdMob 等）。广告推荐的核心技术是推荐引擎，角色画像是广告推荐引擎的一部分。类别：物品信息画像（内容识别、关键字）；用户物品偏好（评价、查看、购买等）；协同过滤推荐；总量性指标；趋势性指标；访问者成本。
- 电商类推荐（京东、淘宝）。根据用户喜好推荐相关用户喜欢的产品。指标：关键字；浏览量；销量；价格；购买喜好；活动推广商品画像。
- 资讯类推荐（今日头条、一点资讯）。根据用户主动订阅或者浏览次数频率，对用户喜好建立画像，进行资讯推荐。指标：订阅；热门推荐；最新推荐；浏览品类次数统计；浏览的频率推荐；关键词推荐。
- 视频类推荐（优酷、YouTube、爱奇艺、腾讯视频等）。视频网站有大量的视频，怎样让用户找到自己感兴趣的视频，需要对用户喜欢的内容做精确的画像，提高用户的体验。指标：基础用户画像；播放历史画像；视频质量画像；视频相关度画像；营销视频画像；用户热点画像。
- 用户金融信用等级画像（芝麻信用分、央行征信记录）。互联网金融大数据，需要对用户信用等级做评估，那么就需要对用户信用画像。指标：固有资产画像；经济能力画像；消费能力画像；朋友圈画像；互联网使用行为画像；标准信用等级画像。
- 电信用户监管系统画像（三大运营商）。指标：用户人口属性画像；用户消费能力画像；用户恶意欠费画像；欺诈用户画像。

虽然用户画像具有如此多的应用领域和场景，但与此同时，我们也需要知道用户画像应用风险点。

用户标签数据的准确与完整性风险

很多时候用户画像的数据来源可靠性无法保证，例如用户自己填写的年龄、性别等信息

往往与其真实情况并不相符，据此进行精准营销很容易适得其反，通常的做法是使用一部分已经标注准确的数据来训练，然后预测用户的人口属性，但这种方式无法评估计算结果的准确性。

目前主流的 Web 浏览器都提供屏蔽第三方的 Cookie 功能，这使得用户在网站上的浏览行为无法被记录下来被二次利用，这就使得数据的完整性完全无法保证。

目前互联网的发展趋势是多屏融合，用户使用各种 PC、笔记本、手机、平板等，这些设备记录下来的用户数据往往不能统一到一起，也就使得企业无法真正完整地了解用户使用互联网的行为，例如苹果和安卓用来识别用户唯一标识的技术就不相同，并且用户还可以手工屏蔽掉广告网络对苹果手机用户的行为追踪。

用户隐私问题

互联网广告是一个典型的个性化系统，它需要大量使用用户的行为数据进行受众定向，同时，在广告市场中还存在着数据交易的产品。无论是受众定向还是数据交易，都需要谨慎地考虑对行为数据的使用是否会泄露用户的隐私；同时也要考虑拥有数据的利益方，特别是广告主，是否在广告市场中被平台或竞争对手获得和利用了自己的关键商业数据。

用户希望其个人隐私受到互联网公司的保护，而事实是由于互联网公司掌握了用户的浏览、购买或者人口属性等信息，而经常给用户推荐其曾经看过、买过、浏览过的商品或者信息造成用户厌倦，或者持续地通过电话、短信、邮件等方式骚扰用户。

目标用户群体不足问题

精准营销有时可能导致广告主发现自己在广告网络上或者媒体中的目标客户群体太小，无法满足市场拓展的需要，尤其对于大的品牌广告主来说，他们希望能尽可能接触到更广的潜在消费群体，例如某生产感冒药的企业，如果只把自己的广告投放给最近患过感冒的人群，显然是荒谬的。

数据安全问题

程序化交易的产生使得在线广告市场可以综合利用需求方和供给方的数据来完成更加精准的广告决策。当然，这样的便利性也是一把双刃剑，在数据得到更加充分利用的同时，RTB 中供给方和需求方对于数据安全性的顾虑和诉求也必须加以考虑。

(1) 供给方数据安全

我们先来看看供给方的数据安全性问题。由于在 RTB 过程中，ADX 需要向参与竞价的 DSP 广播每次展示 URL 和 Cookie，使得 DSP 理论上存在规模化监听媒体用户行为的可能。假设有某个恶意的 DSP 对于能够参与竞价的所有广告请求都以很低的价格参与竞价，目的不在于赢得流量，而在于收集媒体上的用户行为，这就产生了媒体数据的安全问题，我们将其称为供给方数据安全。

供给方的数据安全问题尽管在 RTB 中确实存在，但是并不是想象中那样严重。可以想询价优化技术：由于带宽的限制，实际上在每次询价时，ADX 应该尽可能只向那些最可能赢得竞价的 DSP 发送询价请求，而那些以恶意收集数据为目的的 DSP，在理想情况下应该

被挡在大部分的询价以外。

（2）需求方数据安全

再来看看需求方的数据安全性问题。在 RTB 的环境下，由于定制化标签的引入，广告主的第一方数据也暴露在广告交易的过程中，而这些数据有的是广告主的核心数据，需要认真考虑其安全性问题。为了表达更加清楚，我们用图 9-28 所示的例子来说明。假设有两个英语教育类广告主“英孚教育”和“华尔街英语”，两者都通过 DSP 进行重定向访客找回，那么他们分别利用 RTB 的方式接触到了自己的访客集合。

需要注意的是，这里的消费者集合实际上是广告主的私有数据，也是特别具有商业价值的数数据，然而，DSP、ADX 和媒体都有可能在 RTB 过程中得到这些访客集合。如果 DSP 希望制造更加激烈的竞价环境，获得更高的利润，那么它实际上可以将这两个广告主的消费者集合合并在一起，并生产一个相应的用户标签吸引双方来对此标签竞价。这种做法的实质是在竞争对手之间倒卖消费者集合，并且可以通过比较模糊的标签名字（例如为上面两个广告主的访客集合打上“英语教育”的人群标签）非常隐蔽地操作。随着竞价激烈程度的增加，原本属于广告主的利润就向市场其他环节发生了转移，这个问题就是需求方数据安全性问题。

需求方数据安全性在某种意义上比供给方数据安全性更加重要，因为这决定了广告主是否可以放心地通过 RTB 进行广告采买。坦率地讲，当前的广告交易市场，对这个问题的重视程度和解决方案都还很充分。所以要提醒广告主，在广告交易中使用自己的第一方数据时，特别是面对强势的广告平台时，要特别留意数据安全性的问题。

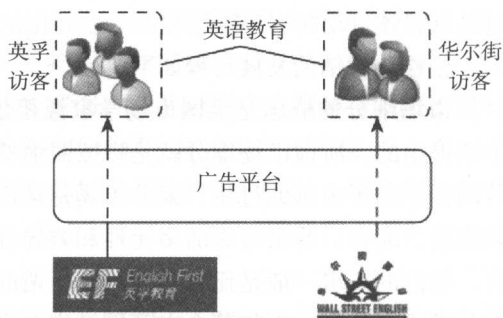


图 9-28 需求方数据安全问题示例

9.3 个性化营销

9.3.1 业务应用背景

所谓个性化营销就是企业面向消费者，通过消费者和企业产品数据库的建立，刻画消费者的画像标签，从而在了解其属性、需求偏好、行为习惯的基础上，直接或间接地了解消费者的需求，在适合的场景下，选择不同的营销载体，通过线下或线上向其推送个性化内容的新营销方式。

随着互联网的发展，电子商务不断扩大，商品种类和服务内容的快速增长，用户需要花费大量的时间才能找到自己感兴趣的物品或内容，由于这些大量浏览无关信息的过程，用户不断地被淹没在冗余的信息和数据加载的过程中，必将导致用户流失的增加及效果转化的降低。为了解决这些问题，提升信息传递的有效性，个性化营销需求应运而生。

个性化营销是根据用户的属性、行为、对象、时间和地域，运用大数据算法刻画出用户的兴趣、习惯偏好和价值度，结合用户体验的制约条件和自身可提供的商品或服务内容，通过消息载体自动向用户推送和展示差异化信息的过程。

个性化营销的发展历程如下：

市场细分的概念是美国市场学家温德尔·史密斯（Wendell R.Smith）于 20 世纪 50 年代中期提出的。所谓市场细分就是指按照消费者欲望与需求把一个总体市场划分成若干个具有共同特征的子市场的过程。细分市场是从消费者的角度，根据市场细分的理论基础即消费者的需求、动机、购买行为的多元性和差异性来划分的。因为在一个垂直领域中同类型的消费者，他们的需求一般是比较相似的，所谓的一个垂直领域在零售电子商务上来说，可以是一个分类或者品类，例如某个电商网站中，有服饰鞋帽、家用电器、手机数码、电脑办公、汽车用品、图书音像、游戏充值和食品酒类等，而在食品酒水中经常浏览或收藏，甚至将红酒加入购物车的消费者，在收入水平、性别、年龄相似的情况下，往往会选择相似价位、相似品牌和相似口感的红酒，这款红酒可能是 60 ~ 80 元、长城（或张裕）、干红。

通过市场细分对企业的生产、营销起着极其重要的作用。传统的市场营销是人以群分，将消费者划分为思想不相同、需要不相同的各个群体。要想使这种方法有效，群体必须足够大，才能盈利。然而，随着消费群体的不断增大，消费者的进一步细分，注定了不断增强的多样性使得为每一群体提供商品或服务的难度增加。各群体需要的范围将会更广泛，社会的分解将使每一人群中有一代表性的人更难找到，差异不断显现，那么确认每一人群的需要的难度将越来越大。这样就使得传统的市场细分难以为继，成功的企业将了解这种新的多样化，并通过更多的相互交流去发现、了解个体的特定需求，而这种特定的需求无疑带有强烈的个人色彩。厂家也许会感觉到这种需求不可理解，但对于某个消费者而言却非常迫切并且也正是这千万种“不可理喻”的不相同的需求才是我们生活的真实世界，也是我们众多企业在网络时代将直接面对的。这样，企业的细分市场将越来越小，或者说每个客户都将成为你的细分市场，个性化营销应运而生。

1995 年 3 月，卡耐基·梅隆大学的 Robert Armstrong 等在美国人工智能协会上提出了个性化导航系统 WebWatcher；斯坦福大学的 Marko Balabanovic 等在同一会议上推出了个性化营销系统 LIRA。

1995 年 8 月，麻省理工学院的 Henry Lieberman 在国际人工智能联合会议（IJCAI）上提出了个性化导航智能体 Litizia。

1996 年，Yahoo 推出了个性化入口 MyYahoo。

1997 年，AT&T 实验室提出了基于协同过滤的个性化营销系统 PHOAKS 和 ReferralWeb。

1999 年，德国 Dresden 技术大学的 Tanja Joerding 实现了个性化电子商务原型系统 TELLIM。

2000 年，NEC 研究院的 Kurt 等为搜索引擎 CiteSeer 增加了个性化营销功能。

2001 年，纽约大学的 Gediminas Adoavicius 和 Alexander Tuzhilin 实现了个性化电子商务网站的用户建模系统 1: 1Pro。

2001年, IBM公司在其电子商务平台 Websphere 中增加了个性化功能, 以便商家开发个性化电子商务网站。

2003年, Google 开创了 AdWords 盈利模式, 通过用户搜索的关键词来提供相关的广告。AdWords 的点击率很高, 是 Google 广告收入的主要来源。2007年3月开始, Google 为 AdWords 添加了个性化元素。不仅仅关注单次搜索的关键词, 而是对用户近期的搜索历史进行记录和分析, 据此了解用户的喜好和需求, 以便更为精确地呈现相关的广告内容。

2007年, 雅虎推出了 SmartAds 广告方案。雅虎掌握了海量的用户信息, 例如用户的性别、年龄、收入水平、地理位置以及生活方式等, 再加上对用户搜索、浏览行为的记录, 使得雅虎可以为用户呈现个性化的横幅广告。

2009年, Overstock (美国著名的网上零售商) 开始运用 ChoiceStream 公司制作的个性化横幅广告方案, 在一些高流量的网站上投放产品广告。Overstock 在运行这项个性化横幅广告的初期就取得了惊人的成果, 公司称: “广告的点击率是以前的两倍, 伴随而来的销售增长也高达 20% ~ 30%。”

2011年8月, 纽约大学个性化营销系统团队在杭州成立载言网络科技有限公司, 在传统协同滤波推荐引擎基础上加入用户社交信息和用户的隐性反馈信息, 包括网页停留时间、产品页浏览次数、鼠标滑动、链接点击等行为, 辅助推荐, 提出了迄今为止最为精准的基于社交网络的推荐算法。团队目前专注于电商领域个性化营销服务以及商品推荐服务社区。

2011年9月, 百度世界大会 2011 上, 李彦宏将推荐引擎与云计算、搜索引擎并列为未来互联网重要战略规划以及发展方向。百度新首页将逐步实现个性化, 智能地推荐出用户喜欢的网站和经常使用的 App。

9.3.2 主要实现过程

个性化营销从宏观上来看包含三个层面: 一是通过线下层面执行; 二是通过线上层面执行; 三是线上和线下执行同样策略的个性化营销, 这就是互联网意义的 O2O (Online To Offline)。而个性化营销的执行和控制是一个相当复杂的过程, 它不仅意味着线下营销人员每个面对消费者的营销人员要时刻保持态度热情、反应灵敏, 更主要也是最根本的是, 它要求能识别、追踪、记录个体消费者的个性化需求并与其保持长期的互动关系, 从而提供个性化的产品或服务, 并运用针对性的营销策略组合去满足消费者的需求。从线上来说, 需要通过流程化的系统软件工作程序, 结合业务规则、大数据算法和各种规则配置引擎来实现。所以, 个性化营销的基础和核心是企业与消费者建立起一种新型的个人学习和机器学习的关系, 即通过与消费者的一次次接触而不断增加对消费者的了解, 利用新型的学习关系, 企业营销人员以及个性化营销系统可以根据消费者提出的要求以及对消费者的了解, 实现线上和线下的联动, 以此提供符合单一消费者特定需要的个性化产品或服务, 最后即使竞争者也进行类似的关系营销, 但你的消费者也不会轻易离开, 因为他还要花费同样甚至更多的时间和精力才能使竞争者对他有同样程度的了解。

由于本书的个性化营销的实现是以互联网为基础的应用，且更多地倾向于互联网电子商务的使用，因此就仅基于互联网线上的角度来阐述个性化营销的实现过程。

首先了解一下个性化营销的整体架构，我们主要以软件系统的角度来进行阐述，从系统角度看个性化推荐架构主要包含数据源、数据仓库、推荐规则引擎、个性化推荐算法引擎、数据挖掘引擎、推荐场景引擎、推荐数据接口、推荐展示引擎和效果优化引擎（如图 9-29 所示）。个性化推荐通过综合并利用用户的属性、网络行为和兴趣偏好，商品/服务的属性、内容和分类，结合时间和场景的关系等，挖掘用户的喜好和需求，主动向用户推荐其感兴趣或者需要的商品和服务内容。

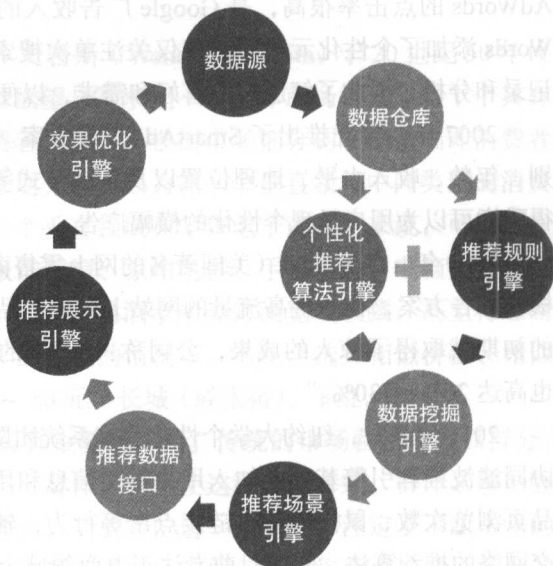


图 9-29 个性化整体流程

上述流程中，数据源和数据仓库不论是线下的营销还是线上的个性化营销推荐都是必须具备的，数据源和数据仓库就像营销系统的子弹，没有它们就谈不上个性化，同时数据仓库也是大数据技术中存储与调用用户画像和商品画像的地方。而对于个性化推荐算法引擎、推荐规则引擎、数据挖掘引擎、推荐场景引擎、推荐数据接口、推荐展示引擎和效果优化引擎等都属于技术层面的内容，不在本章节的范围之内，但为了更好地了解业务实现过程，我们将了解具体的业务规则类型。

1. 零售电商的推荐

说推荐不可能不讲到亚马逊，传言亚马逊有三成的销售额都来自个性化的商品推荐系统。实际上，我们相信大部分消费者愿意主动地去看亚马逊到底给我推荐了什么。一般来说，主流推荐算法是基于一个“跟你喜好相似的人喜欢的东西你也很有可能喜欢”的假设，即协同过滤算法。主要的任务就是找出和你行为、需求偏好相近的消费者，从而根据最近他的行为、需求偏好预测你也可能喜欢什么。

协同过滤算法这种方法可以推荐一些内容上差异较大但又是用户感兴趣的产品，很好地支持用户发现潜在的兴趣偏好，而且这种方法不需要专业领域的知识，例如产品知识、分类知识和行业特殊的知识内容，并且可随着时间推移，自动学习对应的业务规则，从而使性能逐步提高，因此对于新用户以及没有学习基础的营销推荐系统，该推荐算法可能产生质量较差的情况。

关联规则算法也是电商行业常常会使用的推荐类型。即以频繁项集为关联规则基础，把已购的历史商品作为前项规则，即之前购买了什么，而把最后购买的商品作为后项规则，即为被推荐的对象。比如，你购买了手机产品，那么相应地会向你推荐手机周边的配件用品。

关联规则挖掘可以发现不同商品在销售过程中的相关性，在零售业中已经得到了成功的应用。

关联规则过程中，最经常使用的算法是 Apriori 算法，该算法中最核心的配置参数为最低条件支持度、最小规则置信度和最小提升度（如图 9-30 所示）。

图 9-30 关联推荐规则的系统配置界面

- 最低条件支持度：调大该参数，会使参数与分析的前项组合变少，所产生的规则更具普遍性；调小该参数，使得规则变多，但普遍性下降。
- 最小规则置信度：调大该参数，使得规则的预测命中率阈值提高，提升了整体预测命中率，同时规则数量下降，也会导致预测数量下降；调小该参数，则使得规则增多，预测数量增大，但预测精度降低。
- 最小提升度：显示规则置信度与具有结果的先验概率的比，即推荐规则的效果好坏评价标准。

2. 基于内容的新闻资讯类推荐

如果是推荐资讯类的新闻都会采用基于内容的推荐，甚至早期的邮件过滤也采用这种方式，常用的算法为贝叶斯算法（如图 9-31 所示）。基于内容的贝叶斯推荐方法就是根据用户过去的行为记录来向用户推荐相似内容。简单来说，就是你常常浏览科技新闻，那就更多地向你推荐科技类的新闻。

通过内容相关特征的属性来定义项目或对象，系统基于用户评价对象的特征学习用户的兴趣，依据用户资料与待预测项目的匹配程度进行推荐，努力向客户推荐与其以前喜欢的产品相似的产品或内容。

基于内容过滤的系统其优点是简单、有效。尤其对于推荐系统常见的在清空缓存或者完全没有浏览记录不能使用的情况下，该方法

图 9-31 贝叶斯算法的系统配置界面

能够比较好地进行解决。因为该算法不依赖于大量用户的点击日志，只需要使用待推荐对象本身的属性、类目、关键词等特征，因此该方法在待推荐对象数量庞大、变化迅速、积累点击数稀少等应用场景下有较好的效果。但该方法的缺点是对推荐物的描述能力有限，过分细化，导致推荐结果的精度较差，推荐结果往往局限在与原对象相似的类别中，无法为客户发现新的感兴趣的资源，只能发现和客户已有兴趣相似的资源，并且只有维度增加才能增加推荐的精度，但是维度一旦增加计算量也呈指数型增长。同时，由于内容推荐属于虚拟的软推荐，对于推荐内容的格式渲染也不是一件容易的事，同一个作者的排版和风格等都会发生改变。

3. 基于知识类的推荐

自从浏览器 Cookies 和手机的 IMEI 识别号可以获取之后，互联网企业可以通过大数据算法对用户的基本属性、行为、需求偏好等进行预判和预测，尤其在广告领域的个性化投放也可以根据不同场景使用了。

当用户的行为数据较少时，基于知识的推荐可以帮助我们解决这类问题。用户必须指定需求，然后系统设法给出解决方式。假设，你的广告需要指定某地区某年龄段的投放，系统就根据这条规则进行计算。基于知识的推荐在某种程度上可以看成是一种推理技术。这种方法不需要用户行为数据就能推荐，所以不存在清空缓存或者完全没有浏览记录不能使用的情况。推荐结果主要依赖两种形式：基于约束推荐和基于实例推荐。更多关于精准广告的推荐方式和规则，我们将在 9.4 节重点阐述。

9.3.3 关键应用场景

个性化主要是针对现在消费者千差万别的不同需求，但是这个需求不仅限于营销，还可以扩展应用到产品的生产、产品的流通、产品的使用和产品使用反馈的整个供应链过程中。因此，在具备消费者画像数据库和产品画像数据库的基础上，可把个性化进一步分成三个部分，即产品生产个性化、产品服务个性化和营销模式个性化。

1. 产品生产个性化

产品生产的个性化其实是产品最大限度地满足消费者个性化的消费心理诉求，彰显与他人不同的地方，满足人格差异化的表现心理，从而促使企业转向定制化的产品生产，该生产过程包括产品可定制化的结构设计、定制化的外观设计、定制化的形象概念包装、定制化的功能实效性设计和差异化的价格定位等。但鉴于企业生产成本的问题，大部分产品的个性化很难做到与消费者进行“一对一”的定制化，只能是最大限度地接近需求，随着个性化需求的不断涌现，未来有可能通过技术革新和转变，促使企业优化生产工艺，将现在产品的量化生产，变革为工艺模块化、定制化的生产过程，通过多种模块工艺的组合，形成新型的产品个性生产流程，从而最大限度地满足单一消费者的个性需求，同样在产品生产过程实现个性化的同时，产品的价格定位也将迎合不同消费的需求以及消费能力来指定。但这个过程也会存在一个问题，即将同种产品的“个性价格”过于差异化，这样可能会影响高端消费层次的消费心态而失去这一市场。

2. 产品服务个性化

产品服务的个性化是消费者个性化的重中之重,在产品异常丰富的当下,产品无论如何“个性”都摆脱不了被快速同质化的命运,更何况要满足单一消费者的差异化需求,企业成本固然会水涨船高,但是服务的个性化相比产品生产的个性化,可以更加灵活多变,而且更加直接有效的令消费者有直观的感知和感受。

个性化服务直接体现于产品的附加值,是产品增值的重要方式,例如购买裤子可以终身免费干洗,购买自行车终身免费维修和免费打气;再如购买电脑可以终身免费升级操作系统等。随着现在消费者的消费能力日趋提高,消费需求不再限于产品的功能本身价值上,更多地在于消费者购买产品前后的一种心理诉求得到最大满足,但如何让消费者高兴,那就是要开展有效的个性化服务了。

个性化服务是不断满足个人合理需求的差异化服务,个性化服务是非常具体和灵活的服务。对消费者我们必须非常细心地去观察和了解,然后针对性地进行服务,个性化服务必须体现情感价值。对于企业发展个性化服务,必须有稳定的员工队伍,最关键的是要有把握提供优秀的员工。

3. 营销模式个性化

营销模式的个性化主要体现在个性化内容和沟通渠道两个方面,即在消费者画像和产品画像的基础上,结合产品的市场优势,在合理的场景下,选择合适的沟通方式,向消费者传递需要的信息内容,以此满足消费者个性化的需求。

企业选择个性化的营销除了上述内容之外,还需要注意三个原则:一是创新原则,营销模式的每一次创新都代表企业将首先取得成功;二是营销模式的个性化要适应消费者群体的心理认可和生活行为习惯;三是在产品的终端促销方案设计上也是需要迎合消费者的个性化需求,产品的促销只要仔细考虑分析消费者的需求,必能事半功倍。

总之,个性化营销,是营销过程中达成消费者认可、消费者受惠与产品销售成正比结合的具体行为,生产企业和终端销售商均可结合自己的企业性质和产品特点设计个性化营销方案,要注意可行性与落实最重要。

9.3.4 应用价值提炼

个性化消费正在成为市场环境的主要特点,特别是在新生代(90后以及00后)快速成为消费主力的现代化的互联网时代,由此满足不同消费个体的差异化需要的能力,包括个性化的营销和个性化的生产(或服务)能力等是企业生存发展的核心能力,营销特征全面转向个性化,企业需要在消费者的个性化需求和规模效益之间找到最佳契合点。

1. 千人千面的个性化画像

个性化营销较之传统市场营销来说,已由产品差异化转向注重消费者差异化。同时,在消费者群体的差异化,又进一步提升为千人千面一对一的个性化营销。从广义上理解消费者差异化主要体现在两个方面:一是不同的消费者代表不同的价值水平;二是不同的消费者有不同的需求

偏好和习惯。因此，个性化营销认为，在充分掌握了企业消费者的信息资料并考虑了消费者价值的前提下，合理区分企业消费者之间的需求和习惯差异是重要的工作内容，如何区分消费者之间的需求偏好和习惯，主要通过用户画像来实现。消费者差异化对开展个性化营销的企业来说，首先是可以使企业集中有限的资源从最有价值的消费者那里获得最大的收益，令“个性化”营销有的放矢；其次是企业也可以根据现有的消费者信息，重新设计营销行为以及丰富和调整产品结构，从而对消费者的价值需求做出匹配的反应；三是企业对消费者和产品的画像进行标签化管理，定期地更新画像内容，将有助于企业在特定的经营环境下调整合适的经营战略。

2. 建立差异化的沟通方式

面对千人千面的个性化营销，传统的大众媒介已经不再能满足需要，这就要求寻找、开发、利用新的沟通手段。互联网及信息技术的高速发展，为企业与消费者的选择提供了越来越多的“一对一”沟通方式，例如通过精准广告生态系统、微信、APP 消息推送、运营商短消息和邮件的方式，向目标消费者传输及获取最新最有用的信息，这相对一对一的客服电话沟通来说大大节约了成本，但这并不代表可以忽视传统的人员沟通、消费者俱乐部等方式。还有最重要的一点需要强调，沟通方式是根据消费者所在的场景来进行，例如消费者在进行网页浏览时，我们应该在对应的网站，结合其偏好和习惯进行精准广告推荐；再如用户正在查看微信软文内容，我们可以在微信上植入相关性较高的软广告形式，这样既不影响消费者对网站或者消息载体的延误，同时经过实践证明，这种方式的转化效果是目前最好的，能较传统媒体方式的效果提升 3 ~ 5 倍，甚至更高。

3. 企业流程“定制”

个性化营销的最后一步是定制企业流程，经过不断的分析和验证以后再重构，重构的部分应该重点放在供应链、营销、销售和服务流程，将它们进行重新解剖，划分出相对独立的子过程，再进行重新组合，设计各种个性化场景或轻应用的程序，以较低的成本重构各种各样的营销推荐过程以满足消费者的需求；即个性化营销最终实现的目标是为单个消费者提供完全满足其需求的产品和服务内容。

9.3.5 场景总结回顾

个性化营销推荐包含零售电商类的产品推荐、新闻资讯的内容推荐和知识传递的广告推荐，各种推荐都使用了不同的算法，但每种不同的算法都有自己的长处和短处，例如协同过滤方法的缺点有：

- ❑ 稀疏性问题：如果用户对商品的评价非常稀疏，这样基于用户的评价所得到的用户间的相似性可能不准确。
- ❑ 可扩展性问题：随着用户和商品的增多，系统的性能会越来越低。
- ❑ 清空缓存或无记录的新用户问题：如果从来没有用户对某一商品加以评价，则这个商品就不可能被推荐。

不同类型的个性化营销推荐有自己特定的应用场景,但总体来说个性化营销推荐在当今线上和线下的营销过程中都起着越来越重要的作用:

个性化推荐的最大优点在于它能收集用户特征资料并根据用户属性、行为、需求偏好以及习惯,为用户主动做出个性化的推荐,给出的推荐产品或内容还是可以实时更新的,即当产品画像或消费者画像发生改变时,给出的推荐规则也会自动改变,这就极大地提升了互联网以及线下营销的简便性和有效性,同时提高了企业的整体经济效益。

总体说来,一个成功的个性化营销推荐系统的作用主要表现在以下三个方面:

1) 提高将过客转化为消费者的概率:访问者在浏览过程中经常并没有购买欲望,个性化推荐系统能够向用户推荐他们感兴趣的产品,从而促成购买过程。

2) 提高网站销售的整体转化率:个性化营销推荐在消费者购买过程中向其提供更多有针对性的产品或内容,消费者能够从推荐列表中选择自己需要但在浏览过程中没有想到的商品,从而有效提高产品的交叉销售概率。

3) 提高客户对网站的忠诚度:与传统的方式相比,个性化营销推荐使得消费者拥有更多相似和相近的选择,消费者更换网站的成本极低,只需要点击一两次鼠标就可以在不同的站点之间跳转,反之给电商网站带来的成本则极高,因为现在带来一个浏览客户的成本已经在 0.3 ~ 1 元,而从一个浏览客户到一个真正消费的消费,这个成本可能在 80 ~ 200 元(对于不同经营类型和产品丰富度的网站,成本有较大的差异)。个性化营销推荐分析消费者的购买习惯,根据消费者需求向其提供有价值的产品推荐,从而使消费者对个性化形成依赖。

个性化营销推荐具有良好的发展和应用前景。目前,几乎所有的大型电子商务系统,例如 Amazon、eBay 等不同程度地使用了各种形式的推荐系统。在日趋激烈的竞争环境下,个性化营销推荐能有效地保留客户,提高网站的服务能力。

9.4 精准广告

9.4.1 业务应用背景

精准广告指通过大数据技术,将用户进行标签化画像,根据用户当下的需求与企业的产品推广需求进行匹配,在网络系统平台上,通过文字、图片、动画或视频四种形式,精准地向用户投放营销内容的广告形式。

追本溯源,网络广告发源于美国。1994 年 10 月 27 日是网络广告史上的里程碑,美国著名的 Hotwired 杂志推出了网络版的 Hotwired,并首次在网站上推出了网络广告,这立即吸引了 AT&T 等 14 个客户在其主页上发布广告 Banner,这标志着网络广告的正式诞生。更值得一提的是,当时的网络广告点击率高达 40%。

中国的第一个商业性的网络广告出现在 1997 年 3 月,传播网站是 Chinabyte,广告表现形式为 468 × 60 像素的动画旗帜广告。Intel 和 IBM 是国内最早在互联网上投放广告的广告主。中国网络广告一直到 1999 年年初才稍有规模。历经多年的发展,网络广告行业经过数

次洗礼已经慢慢走向成熟。

1999 年 4 月中旬, Doubleclick 派员来京, 与传立、新浪、搜狐洽谈合作。

2000 年 4 月 30 日, 北京广播学院成立网络传播学院, 设网络广告系专业。

2000 ~ 2002 年随着互联网进入寒冬, 网络广告的发展也开始进入蛰伏期。

2001 年中国网络广告市场为 4.1 亿元, 2002 年中国网络广告市场规模为 4.9 亿元。

2003 年春, 非典 (SARS) 突然降临, 在非典 (SARS) 的“帮助”下, 网络广告在 2003 年开始爆发。有数据显示, 2003 年中国网络广告的市场规模急剧增至 10.3 亿元, 增长幅度达 112%。

2004 ~ 2005 年, 由于互联网环境的改变, 众多互联网公司开始盈利, 风投重新大批进入互联网产业, 而网络广告市场也稳步增长, 平均增长率在 70% 以上。

2006 ~ 2007 年, 传统的网络广告模式已经不能满足客户的需求, 于是各种网络广告模式百花齐放, 而网络广告代理公司也成为资本的宠儿。中国排名前两位的网络广告代理公司好耶与华扬联众陆续被收购。

2005 年中国网络广告市场规模为 31.3 亿元, 比 2004 年增长了 77.1%。2006 年中国网络广告市场规模为 46.6 亿元, 比 2005 年增长了 48.9%。2007 年中国网络广告市场规模达到 62 亿元, 比 2006 年增长了 33%; 2008 年网络广告市场受到北京奥运会因素的影响, 网络广告市场规模达到了 98 亿元, 获得了 57% 的增长率。我国网络广告获得了快速的发展。2005 年, 我国网络广告营业额超过了 30 亿元, 2006 年更是攀升至 46 亿元, 2007 年中国网络广告市场规模达到 62 亿元。2009 年第一季度, 中国网络广告市场跌入 2007 年第三季度以来的最低谷。然而从 2009 年第二季度开始, 中国网络广告市场出现强劲反弹, 2009 年市场规模达 207.4 亿元, 年同比增长 22.0%。2010 年市场规模达 105.6 亿元, 年同比增长 51.8%。2011 年中国网络广告市场规模达到 511.9 亿元, 较去年增长了 57.3%。然而 2012 年之后的中国网络广告市场规模及发展趋势预测如图 9-32 和图 9-33 所示。



图 9-32 艾瑞咨询提供的 2012 ~ 2018 年的中国网络广告市场规模及预测

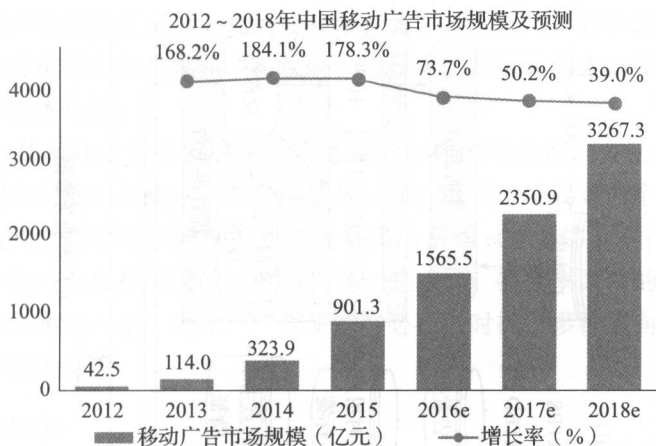


图 9-33 艾瑞咨询提供的 2012 ~ 2018 年的中国移动广告市场规模及预测

9.4.2 主要实现过程

1. 数据收集

建立数据管理平台, 收集用户信息并储存在数据管理平台的结构化数据库中, 用户数据库将现有的、可接近的、可接触的用户或者潜在用户的众多信息, 以组织化的方式汇总成一个系统, 以实现营销或者产品设计的目的。理想的用户数据库, 应该包含用户的注册信息、人口统计信息 (年龄、收入、教育程度、生日等)、询问记录、过往浏览记录、过往购买记录、心理信息 (活动、兴趣、观点)、媒体信息 (喜欢的媒体) 等。

数据管理平台的系统架构如图 9-34 所示。通过部署在媒体上的代码或 SDK 收集第一方访问日志, 送入数据高速公路。同时, 通过数据高速公路收集自有的第二方数据, 然后把这些日志原始行为映射到结构化或非结构化的受众标签体系上。另外, 还会有一些第三方提供的加工好的标签数据直接进入用户标签集。最后通过统一的接口对外提供标签。在这一架构中, DMP 同时对接了第一方、第二方和第三方的数据, 并根据这些数据对受众群体进行灵活的、自定义的划分。虽然这些功能并不直接体现在广告交易环节中, 却是数据驱动的在线广告中越来越重要的一环。

除了需要用到上面讨论的受众定向技术, DMP 还有一个技术问题, 即如何将用户标签传送给购买方, 例如某 DSP。这包括两个环节: 一是用户身份对应, 例如 Cookie 映射; 二是数据的传递方式。

2. 用户数据标签化

用户画像的底层是机器学习, 那么无论是要做用户分群还是精准营销, 都先要将用户数据进行规整处理, 转化为相同维度的特征向量, 诸多算法才可以有用武之地, 例如聚类、回归、关联、各种分类器等。对于结构化数据而言, 特征提取工作往往都是从给数据打标签开始的, 例如购买渠道、消费频率、年龄性别、家庭状况等。好的特征标签的选择可以使对用户的刻画变得更丰富, 也能提升机器学习算法的效果 (准确度、收敛速度等)。

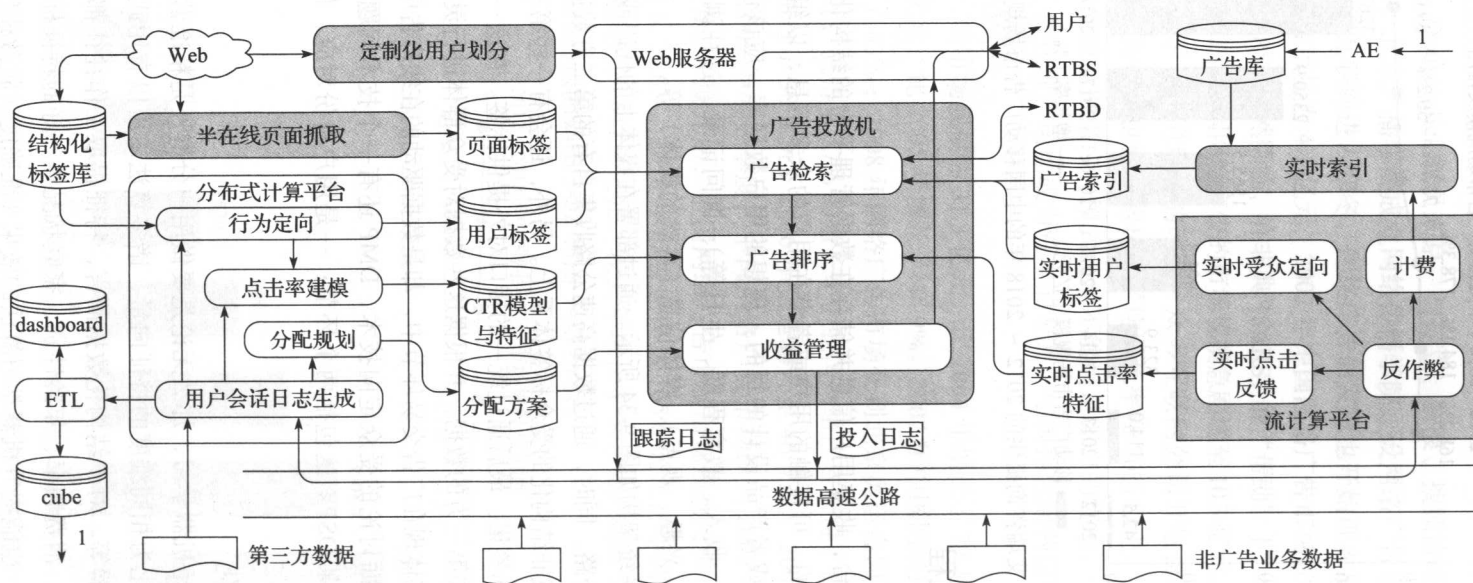


图 9-34 数据管理平台 (DMP) 架构图

受众定向是用户数据标签化的一项核心技术，大多数互联网广告产品的基础是按照受众售卖。因此，受众的标签定向是其非常重要的支持技术，所以我们先对受众定向这一核心的广告产品策略进行整体介绍。

随着在线广告技术和业务的发展，产生了各种各样的受众定向方法，这些方法的综合应用使得广告的精准程度越来越高。在考察某种定向方法时，主要有两个方面的性能需要关注：一是定向的效果，即符合该定向方式的流量高出平均 eCPM 的水平；二是定向的规模，即这部分流量占整体广告库存流量的比例。当然，效果好、覆盖率又高的定向方法是我们追求的目标，不过往往难以两全。因此，广告系统有必要同时提供多种定向方法的支持，以达到整体流量上质的最优化。

受众定向方法概览

在图 9-35 中，水平方向表示的是定向技术在广告信息接收过程中大致起作用的阶段，而垂直方向为定性的效果评价。对受众定向的一些典型方法，我们举例说明如下。

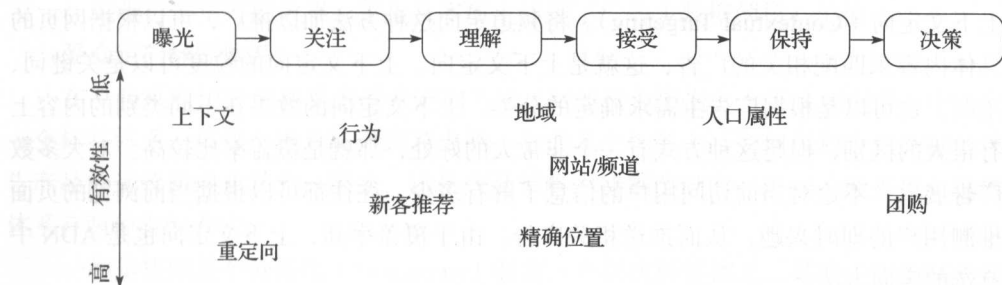


图 9-35 常见受众定向方法一览

□ 地域定向（Geo-targeting）。这是一种很直观也很早就被广泛使用的定向方式。由于很多广告主的业务有区域特性，地域定向的作用相当重要，也是所有在线广告系统都必须支持的定向方式。地域定向也可以认为是一种上下文定向，不过其计算简单，仅仅需要简单的查表就可以完成。地域定向是一种不可或缺的流量选择手段。举个例子，假设某电商网站只在北京运营和送货，那么其效果广告一般来说应该定向在北京的区域内，否则一个其他省的消费者点击广告进入购物环节后，如果发现无法结算，将会是非常差的用户体验。

□ 人口属性定向（Demographical Targeting）。人口属性定向虽然在效果上未必特别突出，但是由于在传统广告的话语体系中大量使用这类标签来表达受众，因此它特别为品牌广告主所熟悉。在线广告的品牌合约中也经常会有对人口属性的要求。人口属性的主要标签包括年龄、性别、教育程度、收入水平等。人口属性有一点与兴趣标签不同，那就是它是可以监测的，即可以用采样加调研的方法来判断一次人口属性定向广告活动受众中有多少比例是正确的。因此，在按 CPM 结算的广告中，人口属性比其他定

向标签为广告主接受的程度更高。

需要说明的是，除非有特别的专门数据来源，例如实名制 SNS 的注册信息或在线购物的消费记录等，一般情况下要进行准确的人口属性定向并不容易。在人口属性数据覆盖率不足的情况下，如果要按照这种定向进行 CPM 售卖，我们可以用已知人口属性的用户作为训练集，构造分类器对人口属性进行自动标注。一般来说，采用分类器的方法确定人口属性准确程度有限。在单纯效果类的广告活动中，预测人口属性的必要性不太高，因为预测出来的人口属性也是根据用户其他行为特征得到的，并不能提供额外的信息量。

- ❑ 频道定向 (Channel Targeting)。频道定向是完全按照供应方的内容分类体系将库存按照频道划分，对各频道的流量投送不同的广告。这种定向方式比较适用于那些离转化需求比较近的垂直类媒体，例如汽车、母婴、购物导航等。对于内容覆盖面比较宽的媒体，这种方式取得的效果是有限的。举一个极端的例子，如果我们把某网站的军事频道作为一个定向标签，那么很难找到直接匹配的广告需求。
- ❑ 上下文定向 (Contextual Targeting)。将频道定向这种方法加以推广，可以根据网页的具体内容来匹配相关的广告，这就是上下文定向。上下文定向的粒度可以是关键词、主题，也可以是根据广告主需求确定的分类。上下文定向的效果在不同类别的内容上有很大的区别，但是这种方式有一个非常大的好处，那就是覆盖率比较高。对大多数广告展示，不论对当前访问用户的信息了解有多少，往往都可以根据当前浏览的页面推测用户的即时兴趣，从而推送相关广告。由于覆盖率高，上下文定向也是 ADN 中首选的定向方法之一。
- ❑ 行为定向 (Behaviorial Targeting)。行为定向是展示广告中非常重要的一种定向方式，其框架是根据用户的历史访问行为了解用户兴趣，从而投送相关广告。行为定向之所以重要是因为它提供了一种一般性的思路，使得在互联网上收集到的用户行为数据可以产生变现的价值。因此，行为定向的框架、算法和评价指标也就奠定了在线广告数据驱动的本质特征，并催生了相关的数据加工和交易的衍生业务。如果把上下文定向看成是根据用户单次访问行为的定向，那么行为定向可以认为是一系列上下文定向的融合结果。因此，上下文定向是行为定向的基础，而且对各种类型的上下文定向都可以有相对应的行为定向方式。比如，地域定向是根据用户当前访问的 IP 来确定地理区域，相应地，也可以根据用户过去一段时间内的访问中最频繁的地理位置来定向，这种方式实际上得到的是更接近于用户的经常居住地，业界有人称其为“where-on-earth”定向。
- ❑ 精确位置定向 (Hyper-local Targeting)。在移动设备上投放广告时，我们有可能获得非常精准的地理位置。比如，利用蜂窝信息或者 GPS，地理定位的精度完全可以达到街区的粒度，如果进一步利用 Wi-Fi、蓝牙等设备的室内定位技术，精度可以进一步达到数米级。这就使得基于精确地理位置的广告成为可能，也使得大量区域性非常强

的小广告主(例如餐饮、美容等)有机会投放精准定位的广告,这已经与传统意义上的地域定向有了质的变化,也成为移动广告最重要的机会之一。在桌面环境中,也有数据提供商(例如 Experian)可以提供根据 IP 信息得出的电脑精确定位,在这些数据的支持下,桌面在线广告也可以进行精确位置定向。

❑ 重定向(Retargeting)。这是一种最简单的定制化标签,其原理是对某个广告主过去一段时间的访客投放广告提升效果。显然,某个广告主的访客是其独有的信息,因此这属于定制化标签。重定向被公认为精准程度最高、效果最突出的定向方式,不过其人群覆盖量往往较小。这是因为,重定向的覆盖投放量是由广告主固有用户的数量和与媒体的重合比例共同决定的。

❑ 新客推荐定向(Look-alike Targeting)。由于重定向的量太小,而且无法满足广告主接触潜在用户的需求,因此不能仅仅依靠它来投送广告。新客推荐定向的思路是根据广告主提供的种子访客信息,结合广告平台更丰富的数据,为广告主找到行为上相似的潜在客户。这一方法的目的是希望在同等用户覆盖比率的情况下,达到比一些通用的兴趣标签更好的效果,这也从实质上体现了广告主数据的核心价值。

受众定向标签体系

在一些反映用户兴趣类的受众定向方法(例如行为定向、上下文定向等)中,我们需要一个标签体系,将每个用户映射到其中的一个或几个标签上。如何规划合理的标签体系对广告产品的运营影响非常大,因此,这是产品策略中特别关键的一环。一般来说,这样的标签体系有两种组织方式:

一种是按照某个分类法(Taxonomy)制定一个层次标签体系,其中上层的标签是下一层的父节点,在人群覆盖上是包含关系。一些面向品牌广告的受众定向往往采用这种结构化较强的标签体系。需要指出,这一体系中的标签是根据需求方的逻辑而制定,某些在媒体方意义很大的分类标签,例如军事等,由于没有明确的需求对应,不宜出现在标签体系中。

另外一种兴趣标签的组织方式,是根据广告主的具体需求设置相应的标签,所有的标签并不能在同一个分类体系中所描述,也不存在明确的父子关系。这种半结构化或非结构化的标签体系往往包含一些比较精准的标签的集合,因而主要适用于多种目标,特别是效果目标并存的广告主的精准流量选择要求。

选择结构化的兴趣标签体系还是非结构化的兴趣标签体系更多的是商业上的决策,主要需要考虑下面两种情形。

❑ 当标签作为广告投放的直接标的时候(包括 CPM 广告及竞价广告中直接可被广告主选择的人群),这些标签既要能够为广告主所理解,又要方便广告主的选择。因此,在这种情形下,结构化的层级标签体系往往是较合理的产品方案,特别是在 CPM 广告中,标签的划分不能过细。这种结构化标签体系的一个典型代表如表 9-3 所示。从表 9-3 中可以看出,这样的标签体系非常易于理解和操作,在面向品牌广告主售卖时较为适用。

表 9-3 Yahoo!GD 受众定向标签体系

一级标签	二级标签
Finance	Bank Accounts, Credit Cards, Investment, Insurance, Loans, Real Estate, ...
Service	Local, Wireless, Gas & Electric, ...
Travel	Europe, Americas, Air, Lodging, Rail, ...
Tech	Hardware, Software, Consumer, Mobile, ...
Entertainment	Games, Movies, Television, Gambling, ...
Autos	Econ / Mid/Luxury, Salon / Coupe /SUV, ...
FMCG	Personal care, ...
Retail	Apparel, Gifts, Home, ...
Other	Health, Parenting, Moving, ...

□ 当标签仅仅是投放系统需要的中间变量，作为 CTR 预测或者其他模块的变量输入时，那么结构化的标签体系其实是没有必要的，应该完全按照效果驱动的方式来规划或挖掘标签，而各个标签之间也不太需要层次关系的约束。这样的标签体系，比较典型的代表是 Bluekai 的标签体系，由于其面向的对象是追求效果或特殊人群定位的广告主，因而组织上的规整性也就让位于效果的精准性。

还有一种特殊的标签形式，即关键词。直接按照搜索或浏览内容的关键词划分人群和投放广告，往往可以达到比较精准的效果。关键词这种标签体系是无层级关系、完全非结构化的，它虽然很容易理解，但并不太容易操作。不过由于搜索广告在整个在线广告中的重要地位，选择和优化投放关键词这样一项专门技术已经发展得相当充分，因此这种标签也是实践中常用的。

3. 用户行为建模

用户行为定向需要进行大规模的数据挖掘，是在线广告中数据利用和变现最重要的计算问题之一。这一问题可以描述为，根据某用户一段时期内的各种网络行为，将该用户映射到某个定向标签上。而行为定向需要考虑用户标签体系、建模方法、特征生成和评测指标等问题。

使用有监督的机器学习算法对用户标签数据与浏览、点击、购买等行为数据进行分析，找出特定用户群标签与特定产品使用行为之间的关联关系，然后输出用户画像的所有标签和对应的关联规则。

通过数据建模，企业可以有效地为能覆盖到的用户打上标签，之后结合渠道信息和商品信息，企业可根据需求定向地选择数据挖掘的方法输出结果，在营销决策中，可能得到的结论如“具有标签 a 的人集中购买了商品 A”“购买商品 B 的用户同样会对商品 A 感兴趣”“商品 A 的购买人群主要集中于渠道 c”等，这些信息将直接指导企业完成营销决策。在这个过程中常用的算法包括聚类和关联规则等，在此不深入展开，这些算法的核心逻辑可以认为是利用现有事实对未来进行预测的过程。

4. 精准广告生态组成

DMP 数据管理平台 (Data Management Platform)：通过全面整合管理各方数据、深度建

模和人群细分,建立自动化人群策略,提供全面深入的数据洞察和智能管理,指导广告主进行广告优化和投放决策。DMP可以帮助广告主实时地梳理和整合多方数据,通过深度挖掘和智能管理,形成基于人群投放并获得更高效果转化的有效指导。

ADX 广告交易平台 (Ad Exchange): 一个开放的、能够将媒体和广告主/广告代理商联系在一起的在线广告市场 (类似于股票交易所)。

DSP 需求方平台 (Demand Side Platform): 为广告主提供跨媒介、跨平台、移动终端的广告投放平台,通过实时数据分析来进行购买、投放广告,并形成报表。DSP 的实现在很大程度上需要有成熟的 Ad Exchange。

ADN 广告网络 (Ad Network): 介于想出售广告资源的 Web 网站、APP 资源与有发布广告需求的广告主之间,一方面帮助媒体将广告位资源按照受众类型进行整合打包出售;另一方面通过行为定向、频次定向、内容定向等技术帮助广告主精准定向目标人群。

SSP 供应方平台 (Supply Side Platform): 帮助媒体主 (在移动端主要是移动站点和 APP) 进行流量分配管理、资源定价、广告请求筛选,使其可以更好地进行自身资源的定价和管理,优化营收。

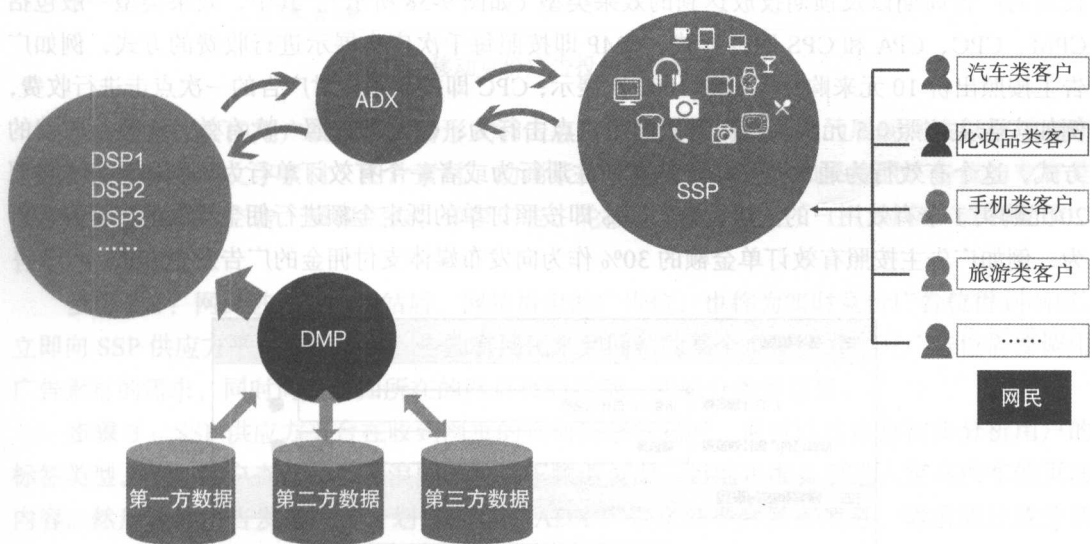


图 9-36 精准广告生态实现流程

步骤 1: DMP 数据管理平台采集企业自有用户数据、合作伙伴数据和网络数据,并对用户数据和产品数据进行画像标签化,存储在数据管理平台中,等待匹配和调用。

步骤 2.1: 广告主 (即广告发布需求方) 在没有自主开发 DSP 需求方平台时,使用外部合作方的平台进行广告发布需求,首先要通过 DSP 需求方平台提交企业相关资质信息,填写企业的相关信息等待审核后,向 DSP 需求方平台进行账户充值。充值完成后便可以开始创建对应的广告计划、广告策略和广告素材 (如图 9-37 所示)。

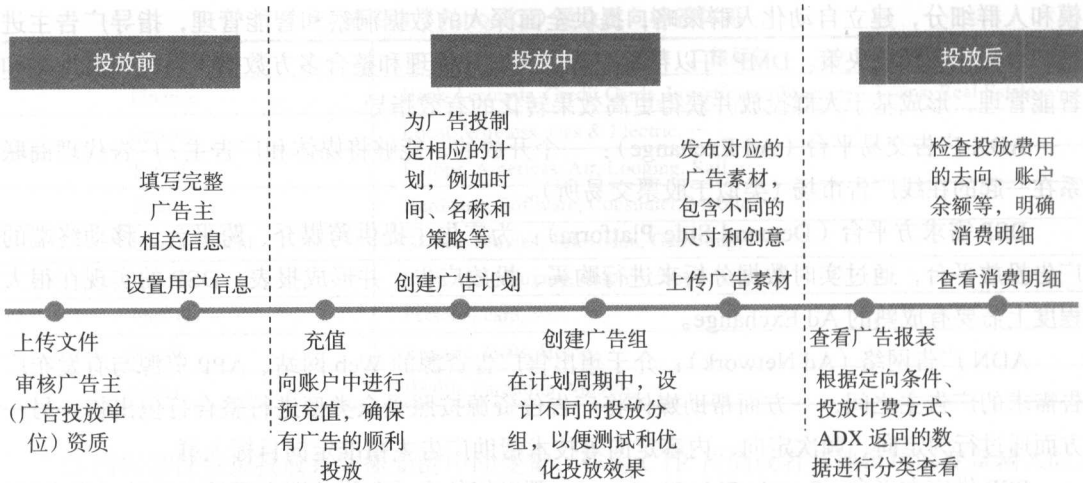


图 9-37 广告主使用 DSP 需求方平台的流程示意

广告计划指广告本次任务总体要花费的预算金额、日预算控制、广告计划任务名称、拟投放的广告周期以及预期投放达到的效果类型（如图 9-38 所示）。其中，效果类型一般包括 CPM、CPC、CPA 和 CPS 四种形式，CMP 即按照每千次广告展示进行收费的方式，例如广告主按照出价 10 元来购买广告的 1000 次展示；CPC 即按照用户对广告的一次点击进行收费，例如广告主按照 0.5 元购买 1 次有效的用户点击行为；CPA 即按照一次有效行为进行收费的方式，这个有效行为通常是指用户的有效注册行为或者一个有效订单行为，例如广告主按照 20 元购买 1 个有效用户的注册行为；CPS 即按照订单的既定金额进行佣金提取的广告发布行为，例如广告主按照有效订单金额的 30% 作为向发布媒体支付佣金的广告发布方式。

The screenshot shows a web-based configuration form for an advertisement plan, divided into two sections: **基本信息 (Basic Information)** and **预算和时间设置 (Budget and Time Settings)**.

基本信息 (Basic Information):

- 广告计划名称:** 请输入广告计划名称 (Text input field)
- 加载已有广告计划设置:** 请选择 (Dropdown menu)

预算和时间设置 (Budget and Time Settings):

- 总预算:** 请输入广告计划总预算 (Text input field) 元
- 每日预算:** 请输入广告计划每日预算 (Text input field) 元
- 投放周期:** 开始时间 (Date picker) 日期不限 更改结束时间 (Text input field)
- 投放目标:** 请选择 (Dropdown menu)

图 9-38 广告计划配置样式示例

广告策略指广告主要展示的用户群体对象特征，主要包括目标用户群体属性定向、地

域定向、媒体定向、时间定向和频次定向等。目标用户群体属性定向如用户性别、年龄、收入、职业等基本信息；地域定向如中国内地的行政区划划分；媒体定向如购物网站、新闻资讯网站、游戏 App、健康医疗 App 等；时间定向如广告发布设定在某一个区间，并在区间内选定既定的时间进行竞价发布；频次定向指广告内容或素材要向同一用户展现的最高频率。频次定向主要也是考虑观看广告的用户体验问题。

移动定向：

APP类别	● 不限 ● 设置
移动网络定向	● 2G ● 3G ● 4G
网络运营商定向	● 移动 ● 联通 ● 电信 ● 其他
移动设备属性定向	● 不限 ● 设置
移动设备品牌定向	● 不限 ● 设置
移动设备操作系统定向	● 不限 ● 设置

图 9-39 移动竞价广告投放系统配置示例

广告素材指要向用户展现的广告形式和内容。广告素材根据不同的展现网站以及对应广告位的类型，分为文字素材、图片素材、动画素材和视频素材等。如果按照广告位的专有类型，可分为文字链广告、横幅广告、弹窗广告、轮播图广告、边栏广告、对联广告和置顶广告等。

步骤 2.2：网民进入某一网站后，网站指定的广告位，也称为实时竞价广告位得到响应，立即向 SSP 供应方平台发布消息，告知有网民来到网站的某个页面，其中有广告位需要调用广告素材的需求，同时向其告知所在的网页页面类型、所属分类等信息。

步骤 3：SSP 供应方平台在收到网页的页面需求信息时，通过这些信息初步分析用户的标签类型，例如用户查看的是新浪网站的汽车频道页面，目前正在准备要进入宝马汽车的页面内容。然后根据广告发布配置计划，立即向 ADX 广告交易平台发布消息，请求通过竞价获得与该用户相匹配的广告素材内容。

步骤 4：ADX 广告交易平台在获取 SSP 供应方平台提供的请求信息后，立即向符合条件的 DSP 需求方平台发布竞价广播，并将竞价条件，即 SSP 供应方平台需要的广告素材条件推送到 DSP 需求方平台。

步骤 5.1：DSP 需求方平台在接到广播消息后，立即将取得的用户标签化内容与 DMP 数据管理平台信息进行快速匹配，查看是否是已有的老用户，还是需要获取的新用户，从而勾勒更加全面的用户画像信息，结合当下用户查看页面的需求进行权重配置，将信息返回 DSP 需求方平台中。

步骤 5.2: DSP 需求方平台在收到 DMP 的画像信息返回后,快速比对广告主在 DSP 需求方平台中发布的广告需求,匹配是否有需要参与本次竞价的广告素材内容,即广告主在 DSP 需求方平台上发布的广告策略内容,如果达到匹配标准,则向 ADX 广告交易平台反馈消息,告知本次参与竞价的价格。

步骤 6: ADX 广告交易平台在收到多个 DSP 需求方平台的竞价信息后,立即进行竞价排名(竞价排名方式不限于 DSP 需求方平台中广告主的出价金额,还可能包含其他因素,例如该 DSP 需求方平台过往竞价成功的概率、响应速度等多重因素),竞价胜出的 DSP 需求方平台,ADX 立即向其发布胜出的通知,并请求调用平台内的广告素材。

步骤 7: DSP 需求方平台在收到 ADX 广告交易平台的调用素材请求后,立即返回相关素材信息。

步骤 8: ADX 收取到素材后直接将素材返回到 SSP 供应方平台上,SSP 立即将素材内容向网民进行展现。

步骤 9: 广告在进行展示后,获取的用户浏览信息、点击信息、跳转信息和购买信息等,结合因为广告的发布和展示产生的费用,一并在 DSP 需求方平台向广告主即广告发布者进行展示,并定期进行财务结算。

整个精准广告的发布流程经过了上述的 9 个步骤,其中从步骤 2.2 开始,直到步骤 8,行业的响应标准应该是控制在 200 毫秒以内(即人体眼睛反应的时间下限,目的是令用户对于整个感知过程,或者叫体验达到可接受的时间范围),这对于硬件、网络和大数据的技术要求速度之快,质量之高是可想而知的。

上述整个实现过程主要是基于广告主发布的流程来进行说明,其中还涉及精准广告生态链中的环节,包括技术细节不在本章节的讨论范围之内。同时,上述广告发布过程根据不同的企业、不同的技术团队和不同的需求,流程上会有所差异。

9.4.3 关键应用场景

随着网络广告市场规模的不断增长,传统网络广告(即展现广告)已经不能满足广告主(即有投放广告需求的企业)的要求,经过互联网广告运营者的探索,逐渐衍生出定向广告,所谓定向广告需要满足两个要求,一是受众定向,即通过技术手段标定某个用户的性别、年龄和其他标签;二是广告投放,即将广告投放由直接嵌入的页面变成实时响应前端请求,并根据用户标签自动决策和返回合适的广告创意。此时的定向广告仍然以合约的方式进行,媒体向广告主保证某个投放量,并在此基础上确定合同总金额以及投放量未完成的赔偿方案。

但这种广告仍然存在两个难题:一是如何有效地将流量分配到各个合约相互交叉的人群中;二是要在在线的环境下实时地完成每一次展示决策。同时在受众定向产生以后,市场向着精细化运作的方向快速发展,这一发展主要有两方面的趋势:一是定向标签变得越来越精准;二是广告主的数量不断膨胀。在这些趋势下,仍然按照合约的方式售卖广告会遇到越来越多的麻烦。因此,合约量的约束带来了麻烦,那么有没有可能抛弃合约量的保证而采用最

唯利是图的策略来进行广告投放决策呢？在这种思路的下催生了革命性的产品模式，即竞价广告，这种模式是攻击方只向广告主保证其单位流量的成本，但不再以合约的方式给出量的保证，换言之，对每次展示都基本按照收益最高的原则来决策。基于大数据的竞价机制和精准人群定向这两个核心功能的结合，从而实现了精准广告。

9.4.4 应用价值提炼

精准广告主要以搜索和程序化广告为主，而程序化购买广告是通过数字化、自动化、系统化的方式改造广告主、代理公司、媒体平台，进行程序化对接，帮助其找出与受众匹配的广告信息，并通过程序化购买的方式进行广告投放，实时反馈投放报表。程序化购买把从广告主到媒体的全部投放过程进行了程序化投放，实现了整个数字广告生态的自动化（如图 9-40 所示）。

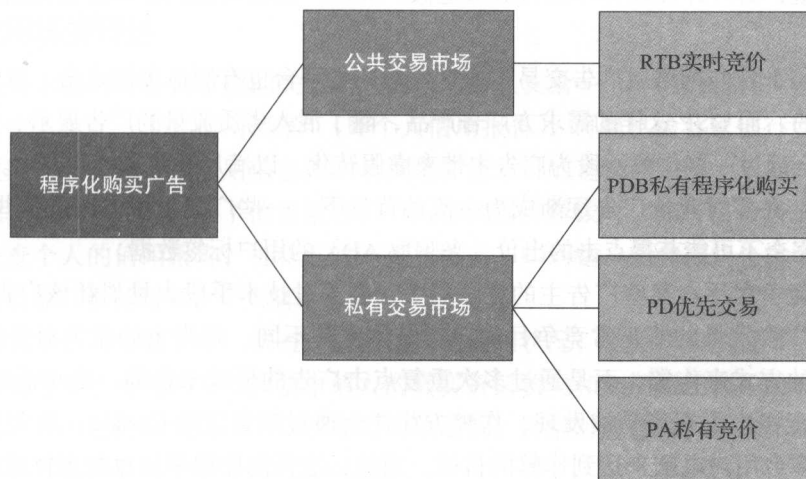


图 9-40 程序化购买广告分类示意图

程序化购买按照交易是否公开可以分为公开交易和私有交易，公开交易主要以 RTB 实时竞价模式为主；而私有交易主要包括三种竞价方式：PDB 私有程序化购买、PD 优先交易、PA 私有竞价，三者的区别在于是否竞价以及广告位是否预留。

- RTB 实时竞价 (Real Time Bidding)：是 DSP、广告交易平台等在网络广告投放中采用的主要售卖方式，在极短的时间内通过对目标受众竞价的方式获得该次广告展现。仅限于购买广告交易资源，无论在 PC 端或是移动端均可以实现 RTB 的购买方式。
- PDB (Programmatic Direct Buying)：私有程序化购买，是把广告主常规购买的保量的优质媒体资源，利用程序化购买的方式进行人群定向等多维度定向的广告投放。无论 RTB 或 PDB 模式都需要 DSP 的系统作为实现投放的桥梁。
- PD (Preferred Deals)：优先交易，与 PDB 的区别在于广告资源具有一定的不确定性，

广告位的展示量，不能预先保证。

- PA (Private Auction)：私有竞价，媒体把较受广告主欢迎的广告位置专门拿出来，放到一个半公开的市场中进行售卖，供有实力的广告主竞价，价高者得。

9.4.5 场景总结回顾

整个程序化广告过程对于广告业的发展是具有变革性意义的生态过程，适应了当下高速发展的经济条件下，用户不断变化的个性化需求，也适应了要求主张自我个性的用户倾向，但程序化精准广告还是存在着很多需要监管和规避的问题，主要包括两个方面：

(1) 广告网络、媒体、广告主在广告投放中的作弊行为

媒体：在互联网广告的投放过程中，广告主、广告网络、媒体三方都有作弊的可能，其中作弊动力最强的显然是媒体，因为虚假流量能够帮助媒体带来更多的收益，但严重损害了广告主的利益，另外即使媒体不进行流量造假，其用户标签数据的准确性也无法保证，正如第1点所说。

广告网络：广告网络或广告交易市场这样的广告平台也有制造虚假点击，以达到获取更多分成的目的。而DSP这样的需求方广告产品，除了混入劣质流量的广告展示、制造虚假点击以外，还会通过一些作弊手段为广告主带来虚假转化，以满足效果考核的要求。

广告主：在实时竞价广告逐渐成为主流的背景下，一些广告主可以通过某些技术手段，例如极低的完全不可能获得点击的出价，来骗取ADX的用户标签数据。

广告主竞争对手：某些广告主的竞争对手，会通过技术手段大量消耗该广告主的预算，达到降低其广告效果的非正常竞争目的。与媒体作弊不同，广告主的竞争对手很难通过控制广告展示的方式来作弊，而是通过多次重复点击广告的形式来作弊。由于通过单一IP或Cookie大量点击广告很容易被发现，作弊方往往会通过频繁清除Cookie，改变IP，甚至通过木马控制多台用户电脑来达到作弊的目的，当然，这样的作弊手段也被媒体或广告平台所采用。单一IP或Cookie在大量展示或点击的作弊方式是最容易去除的，只需要一定时间段内的展示或点击设定合理的上限，进而发现那些显著超过上限的IP或Cookie并加入将这些用户的标签记为黑名单用户即可。

(2) 用户隐私及数据安全问题

大数据浪潮和营销新技术的碰撞，使企业获得了更多更具体的关于客户和他们行为历史的数据。这些信息为市场营销人员分析客户，进行更有效和个性化的营销提供了技术支撑。但是伴随着越来越多的企业开始享受大数据的福利，企业也可能触及到用户以及舆论对保护信息安全和个人隐私的敏感神经。尽管我国还没有制定专门性的个人数据保护法，但对于个人数据保护的法规散落在《宪法》《民法通则》《刑法》《侵权责任法》《互联网信息服务管理办法》《关于加强网络信息保护的决定》等法律中。日前，国务院印发的《关于促进大数据发展的行动纲要》特别提出，要强化信息安全保障，完善产业标准体系。但是企业主动加强对于消费者信息安全及隐私的保护仍然至关重要。据Neustar的调查报告显示，63%的消费者不

会再相信那些泄露过数据的品牌；数据泄露一年后仍有超过 50% 的消费者认为对品牌有负面影响；1/2 的消费者认为安全性和隐私保护对品牌认知来讲很重要。

数据推动时代发展是大势所趋，企业营销中如何保障既能利用用户数据，又能规避用户隐私风险呢？虽然大多数企业分析客户数据并非出自恶意，甚至可以说是为了让客户获得更好的体验。但是作为企业，依然需要在做大数据的挖掘、分析和使用时，把保护数据安全和用户隐私纳入营销管理范畴。而对于市场营销人员，我们需要具备很强的消费者信息安全保护意识，并了解用户在信息安全上的诉求点，在数据的收集和分析方面具备法律意识和专业知识，在营销实践中，企业也需要向客户传递它们是如何保障数据安全、合理透明使用数据的，并且在数据上给予用户充分的知情权和选择权。

9.5 征信

9.5.1 应用场景背景

征信是各家银行将各自掌握的关于个人的信用信息交给一个专门的机构汇总，由这个专门的机构给个人建立一个“信用档案”（即个人信用报告），再提供给各家银行使用。这种银行之间通过第三方机构共享信用信息的活动就是征信，目的是提高效率，节省时间，快点办事。有了征信机构的介入，有了信用报告，个人在向银行借钱时，银行信贷员征得本人同意后，可以查查个人的信用报告，再花点时间重点核实一些问题，便会很快告诉个人银行是否提供给本人借款。银行省事，个人省心。

征信最基本的功能是了解、调查、验证他人的信用状况，使赊销和信贷等活动中的授信人能够较为充分地了解受信人或授信申请人的真实资信状况和如期偿还债务的能力，使授信人的风险降到理论上的最低点。从行业分工的角度看，征信对应着有关信息产品的收集、加工和生产，而不强调信用管理咨询或顾问服务。

中国人民银行对征信的权威定义：征信即征集信用信息，是指对个人、企业的信用信息进行收集、整合，向社会有关方面提供信用信息查询或者增值服务。

个人征信的定义：是指由征信机构把分散在授信机构、司法机关、行政机关等社会各方面个人信用和信誉信息汇集起来，进行加工和存储，形成个人信用信息集合，以此对消费者个人的资信状况进行调查和评估，并在法律允许的范围内向社会提供信用资料和有关信息。当个人在进行个人信用活动时，其信用记录将作为一项重要的参考资料被授信人所考虑，为授信人系统地了解受信人或授信申请人的信用和信誉状况提供服务。

综合前人的定义，个人征信体系可以概括为由一整套个人征信制度、机构、市场、业务、方法、标准、产品、管理和研究等方面构成的系统，是个人征信全部活动的总称。个人征信体系由个人征信的一干法律法规、个人征信服务机构、个人信用信息数据库、个人隐私保护机制、个人征信行业监管、个人信用产品市场、个人信用宣传教育体系等构成，其中组织模式、立法建设、监管机制和隐私保护是最重要的环节。

9.5.2 主要实现过程

1. 数据采集

多维度的征信大数据可以使我们能够不完全依赖于传统的征信体系，对个人信贷者从不同的角度进行描述和进一步深入地量化信用评估。海量信用数据的采集是征信的前提条件，这种数据来源是多元化的，其中可以包括最权威的央行、工商局等政府部门的个人征信、企业注册等数据，也可以包括著名的商业公司如阿里巴巴的芝麻信用分、腾讯微信的用户关系链等数据，还可以包括从第三方获取的数据如法律记录、搬家次数等数据。

征信数据包含但不限于以下内容：

- 个人征信：个人基本身份信息、个人职业信息、个人消费信息、个人信贷信息、个人资产信息、个人社交信息、个人网络行为信息、个人通信信息和其他机构的信用评级等。此外还有网络数据，例如 IP 地址、浏览器版本甚至电脑的屏幕分辨率，这些数据可以挖掘出用户的位置信息、性格和行为特征，有利于评估信贷风险。此外，社交网络数据也是大数据征信的重要数据源。
- 企业征信：企业工商信息、企业经营范围信息、企业交易信息、企业信贷信息、企业资产负债信息、企业现金流信息、企业纳税情况、法院裁决和执行信息、关联企业信息和其他机构的信用评级等。

鉴于各机构的开放程度不同，因此所拥有的信息类型和内容都不一样，但在互联网和大数据技术的驱动下，各家机构的征信信息孤岛被快速地连接和应用，同时信息的查询、使用范围和使用速度越来越快，从目前中国的征信业发展速度来看，逐步在向国外靠拢，未来可能我们的信用信息将是完全透明的。

2. 机器学习

在融合了各类渠道的信息之后，我们采用机器学习的预测模型和集成学习的策略，进行大数据挖掘，对用户进行最常用的信用评分。首先，数千种来源于第三方（例如电话账单和租赁历史等）和借贷者的原始数据将被输入系统。其次，寻找数据间的关联性并对数据进行转换。再次，在关联性的基础上将变量重新整合成较大的测量指标，每一种变量反映借款人的某一方面特点，例如诈骗概率、长期和短期内的信用风险和偿还能力等。然后将这些较大的变量输入到不同的数据分析模型中。最后，将每一个模型输出的结论按照模型投票的原则，形成最终的信用分数。

以征信行业著名的大数据公司 ZestFinance 为例，他们开发了超过 10 个基于机器学习的分析模型，对每位信贷申请人的超过 1 万条的数据信息进行分析，并得出超过 7 万个可对其行为做出测量的指标，在 5 秒钟内就能全部完成。这 10 个模型以如下方式进行投票：让你最聪明的 10 个朋友坐在一张桌子旁，然后询问他们对某一件事情的意见。这种机制的决策性能远远好于业界的平均水平。

近年来，这种基于大数据的信用风险评估框架（远不能称为主流的信用评估方法）被国

内外多家互联网金融机构采用,例如德国的 Kreditech、美国的 Kabbage,以及国内的闪银等,这些对传统的信用体系形成了冲击。

3. 模型输出

有了模型之后,对于征信需求方来说可以通过开发接口程序的方式接入第三方征信机构或评分机构的数据来补充自己的数据来源,从而让各行各业的公司在与用户打交道时都能够提前获知用户的信用得分,能够放心地向优质用户提供借贷、赊销等服务,并且避免向历史信用记录较差的用户提供信贷服务,从而显著降低这些公司的经营风险和坏账率。常用的征信模型包括 Z 计分模型、巴萨利模型、A 值模型、FICO 个人信用评分模型。

Z 计分模型

Z 计分模型是由美国的爱德华·奥特曼教授早在 1968 年研究建立的。它是通过对“健康”企业和“失败”企业样本数据的分析而构建的,它运用关键的财务比率计算出 Z 值,并据此预测公司破产的可能性。

(1) 具体步骤

首先选择能够把健康企业和失败企业区分开的指标;然后计算每一个指标的系数,从而构建模型,计算出 Z 值。模型的具体公式是:

$$Z = c_1x_1 + c_2x_2 + c_3x_3 + \dots + c_nx_n$$

式中:

$x_1, x_2, x_3, \dots, x_n$ ——模型选用的指标;

$c_1, c_2, c_3, \dots, c_n$ ——每个指标对应的系数。

不同性质的企业, Z 计分模型的具体形式也不同。下面介绍三种形式的 Z 模型,其中 Z1 为最基本的模型,适用于上市公司; Z2 适用于非上市公司; Z3 适用于非制造企业。

$$Z1 = 1.2x_1 + 1.4x_2 + 3.3x_3 + 0.6x_4 + 0.999x_5$$

式中:

$x_1 = (\text{流动资产} - \text{流动负债}) / \text{资产总额}$

$x_2 = \text{未分配利润} / \text{资产总额}$

$x_3 = (\text{利润总额} + \text{利息支出}) / \text{资产总额}$

$x_4 = \text{权益市场值} / \text{负债总额}$

$x_5 = \text{销售收入} / \text{总资产}$

Z 值越低,企业信用风险越大,当 $Z < 1.81$ 时,企业信用风险较大;当 $Z \geq 2.99$ 时,企业信用状况良好。

$$Z2 = 0.717x_1 + 0.847x_2 + 3.107x_3 + 0.420x_4 + 0.998x_5$$

式中:

$x_1 = (\text{流动资产} - \text{流动负债}) / \text{资产总额}$

$x_2 = \text{未分配利润} / \text{资产总额}$

$x_3 = (\text{利润总额} + \text{利息支出}) / \text{资产总额}$

$x_4 = \text{所有者权益} / \text{负债总额}$

$x_5 = \text{销售收入} / \text{总资产}$

一般而言，当 $Z < 1.23$ 时，企业信用风险较大；当 $Z \geq 2.9$ 时，企业信用状况良好。

$Z_3 = 6.56x_1 + 3.26x_2 + 6.72x_3 + 1.05x_4$

式中：

$x_1 = (\text{流动资产} - \text{流动负债}) / \text{资产总额}$

$x_2 = \text{未分配利润} / \text{资产总额}$

$x_3 = (\text{利润总额} + \text{折旧} + \text{摊销} + \text{利息支出}) / \text{资产总额}$

$x_4 = \text{所有者权益} / \text{负债总额}$

Altman 认为，根据上述 Z_3 公式计算的 Z 值，如果 $Z < 1.23$ 时，风险很大； $Z \geq 2.9$ 时，风险较小。

另外，Altman 还认为 Z 计分模型开创了资信分析的新思路。它虽不能准确预测公司破产的具体时间，但它指出了破产的可能性，并能通过逐年比较反映出这种可能性扩大或缩小的趋势。1977 年 Altman 又建立了第二代模型，称为 Zeta 信用风险模型。主要变量有 7 个，即资产报酬率、收入的稳定性、利息倍数、负债比率、流动比率、资本化比率、规模。Zeta 模型适应范围更宽，对信用不良客户的辨认精度也大大提高。

(2) Z 计分模型的局限性

- ❑ 主要依赖财务报表的账面数据而忽视资本市场指标，因此，削弱了预测结果的可靠性和及时性。
- ❑ 缺乏对违约和违约风险的系统认识，理论基础比较薄弱，难以令人信服。
- ❑ 假设各指标变量中存在线性关系，而现实的经济现象是非线性的，因而也降低了预测结果的准确程度，使得违约模型不能精确地描述经济现实。
- ❑ 无法计量企业的表外信用风险，另外对某些特定行业的企业如公用企业、财务公司、新公司以及资源企业不适用，因而它的使用范围受到了较大限制。
- ❑ 权数难于确定，权数的精确受到很多因素的影响，这些因素难以量化确定。

巴萨利模型

巴萨利模型是后人在 Z 计分模型的基础上发展出来的。该模型以其发明者亚历山·巴萨利的名字命名，它适用于所有行业，其特点是形式比较简单且不需要复杂的计算。该模型需要用到以下几个比率：

❑ $(\text{税前利润} + \text{折旧} + \text{递延税}) / \text{流动负债}^{\ominus}$

❑ $\text{税前利润} / \text{营运资本}$

[⊖] 流动负债包括银行借款、应付税金、租赁费用。

- 股东权益 / 流动负债
- 有形资产净值 / 负债总额
- 营运资本 / 资产总额

上述5项比率的总和便是该模型的最终指数。指数高说明企业运营状况良好, 实力强; 指数低或负数均表明企业前景不佳。每项比率所对应的指标如下:

- 衡量公司业绩, 能精确计算出公司当前利润和短期优先债务的比值。
- 用于表示公司业绩, 它衡量的是营运资本回报率。
- 是资本结构比率, 衡量股东权益对流动负债的保障程度。
- 衡量扣除无形资产后的净资产对全部债务的保障程度。
- 衡量流动性, 表示营运资本或流动资产净值占总资产的比重。

与Z计分模型相比较, 巴萨利模型适用范围更宽, 被广泛应用于美国金融机构的客户信用分析中, 据调查, 巴萨利模型的准确率可达到95%。巴萨利模型的最大优点是易于计算, 并且它比Z计分模型多一个功能, 即在预测公司破产可能性的同时, 还能衡量公司实力大小, 模型指数高即说明公司实力强, 反之则弱。

A 值模型

在Z计分模型和巴萨利模型中, 财务比率都是依靠企业公开报表提供的数据计算出来的。而处于财务困境之中的企业往往会利用各种手段粉饰企业的财务报表, 仅仅根据财务报表数据计算的结果往往不能反映企业的真实情况, A值模型克服了这一缺陷。A值模型由约翰·阿根提于《企业破产》一书中首先提出, 是一种以更客观的判断为基础的企业破产预测模型。它不仅仅计算财务比率, 还考虑了管理不善可能导致企业破产的因素。

阿根提认为管理不善会导致企业破产, 他在A值模型中列出了管理不善的所有现象, 并将所有这些现象分为管理缺陷、有管理缺陷企业易犯的错误和管理缺陷企业的恶化征兆3个部分共17种, 并给第一种现象制定了最高分值。测试企业时, 根据企业表现打出相应的分数。在总分为100分的测试中, 如果企业总得分超过25分, 那么该企业就有潜在的破产风险。

FICO 个人信用评分模型

FICO 评分系统在美国得到了广泛的使用, FICO 评分系统有五类主要影响因素: 客户的信用偿还历史、信用账户数、使用信用的年限、正在使用的信用类型、新开立的信用账户。

一般来讲, 美国人经常谈到的信用得分, 通常指的是你目前的FICO分数。实际上, FairIsaac公司开发了三种不同的FICO评分系统, 三种评分系统分别由美国的三大信用管理局使用, 评分系统的名称也不同。

FICO 评分系统得出的信用分数范围在300 ~ 850分, 分数越高, 说明客户的信用风险越小, 但是分数本身并不能说明一个客户是好还是坏, 贷款方通常会将分数作为参考, 来进行贷款决策, 每个贷款方都会有自己的贷款策略和标准, 并且每种产品都会有自己的风险水平, 从而决定了可以接受的信用分数水平。一般地说, 如果借款人的信用评分达到680分以

上，贷款方就可以认为借款人的信用卓著，可以毫不迟疑地同意发放款；如果借款人的信用评分低于 620 分，贷款方或者要求借款人增加担保，或者干脆寻找各种理由拒绝贷款；如果借款人的信用评分在 620 ~ 680 分，贷款方就要作进一步的调查核实，采用其他的信用分析工具，作个案处理。

（1）信用偿还历史

影响 FICO 得分的最重要的因素是客户的信用偿还历史，大约占总影响因素的 35%。支付历史主要显示客户的历史偿还情况，以帮助贷款方了解该客户是否存在历史的逾期还款记录，主要包括：①各种信用账户的还款记录，包括信用卡（例如 Visa Master Card American Express Discover）、零售账户（直接从商户获得的信用）、分期偿还贷款、金融公司账户、抵押贷款。②公开记录及支票存款记录，该类记录主要包括破产记录、丧失抵押品赎回权记录、法律诉讼事件、留置权记录及判决。涉及金额大的事件比金额小的事件对 FICO 得分的影响要大；同样的金额下，越晚发生的事件要比早发生的事件对得分的影响大。一般来讲，破产信息会在信用报告上记录 7 ~ 10 年。③逾期偿还的具体情况包括：逾期的天数、未偿还的金额、逾期还款的次数和逾期发生时距现在的时间长度等。比如，一个发生在上个月的逾期 90 天的记录对 FICO 得分的影响会大于一个发生在年前的逾期 90 天的记录。据统计，大约有不足 50% 的人有逾期 30 天还款的记录，大约只有 30% 的人有逾期 60 天以上还款的记录，而 77% 的人从来没有过逾期 90 天以上不还款的记录，仅有低于 20% 的人有过违约行为而被银行强行关闭信用账户。

（2）信用账户数

该因素仅次于还款历史记录对得分的影响，占总影响因素的 30%，对于贷款方来讲，一个客户有信用账户需要偿还贷款，并不意味着这个客户的信用风险高。相反地，如果一个客户有限的还款能力被用尽，则说明这个客户存在很高的信用风险，有过度使用信用的可能，同时也就意味着他具有更高的逾期还款可能性。该类因素主要是分析对于一个客户，究竟多少个信用账户是足够多的，从而能够准确反映出客户的还款能力。

（3）使用信用的年限

该项因素占总影响因素的 15%。一般来讲，使用信用的历史越长，越能增加 FICO 信用得分。该项因素主要指信用账户的账龄，既考虑最早开立的账户的账龄，也包括新开立的信用账户的账龄，以及平均信用账户账龄。据信用报告反映，美国最早开立的信用账户的平均账龄是 14 年，超过 25% 的客户的信用历史长于 20 年，只有不足 5% 的客户的信用历史小于 2 年。

（4）新开立的信用账户

该项因素占总影响因素的 10%。在现今的经济生活中，人们总是倾向于开立更多的信用账户，选择信用购物的消费方式，FICO 评分系统也将这种倾向体现在信用得分中。据调查，在很短时间内开立多个信用账户的客户具有更高的信用风险，尤其是那些信用历史不长的人。该项因素主要包括：①新开立的信用账户数，系统将记录客户新开立的账户类型及总数；

②新开立的信用账户账龄；③目前的信用申请数量，该项内容主要由查询该客户信用的次数得出，查询次数在信用报告中只保存两年；④贷款方查询客户信用的时间长度；⑤最近的信用状况，对于新开立的信用账户，若及时还款，会在一段时间后提高客户的 FICO 得分。

(5) 正在使用的信用类型

该项因素占总影响因素的 10%。主要分析客户的信用卡账户、零售账户、分期付款账户、金融公司账户和抵押贷款账户的混合使用情况，具体包括：持有的信用账户类型和每种类型的信用账户数。

9.5.3 主要应用场景

征信的主要应用场景可分为 3 类：金融征信、行政管理征信、商业征信。

1. 金融征信

金融征信是以金融业主管部门为主导进行建设，以金融机构为主要用户，以授信申请人为主要征信对象，以信用信息在金融业内互联互通、共同防范信用交易风险为主要目的的金融业征信系统及信用管理运行机制的总称。

金融征信的特点：行业征信，准公共征信。我国金融征信体系主要采集的是金融机构传递的信用信息，主要的服务对象也是金融机构。这样的情况就是行业征信，即征信是在行业内部进行的，征信的结果也主要是为本行业服务。作为一个行业信用体系，金融征信体系首先要在金融业内进行信息的共享，其次才是有选择地以有偿或者无偿的方式对外公开一些数据与信息，而公开这些信息的前提是不影响金融行业的安全。目前，我国金融体系主要以银行业为主导，而银行业主要以国有控股为主导。整个金融征信涉及银行、证券、保险、信托等，表现出多层次与多元化的复杂性，集国有经济与市场化之大全，使得金融征信体系具有了准公共征信的特点，一方面要秉承银行为广大社会公众服务的公共性，保护投资者的利益，保护国家金融安全；另一方面也要考虑到股东的利益以及信用信息的市场需求与价值。

2. 行政管理征信

行政管理征信是以政府及其主要职能部门为主导进行建设，以政府及其各职能部门为主要用户，以企业和个人为征信对象，以信用信息在政府及其各部门间互联互通、实现统一的信用惩戒与预警监管为主要目的的政府行政管理征信系统及信用管理运行机制的总称。

行政管理征信的最大特点是公共性，为政府综合信用监管服务。行政管理征信体系事实上是以电子政务为基础，以信用信息整合为切入点，实现政府及其职能部门之间信用信息的共享，形成反映企业和个人综合信用状况的基础数据，实现综合的、有针对性的、预先的监督与管理，以促进社会成员遵纪守法、诚信经营。

行政管理征信中的信用信息系统建设是政府的内部工作，只要符合有关行政管理规定即可，只为政府服务，不需市场化，也不参与市场。该征信体系主要用于政府部门对市场的监管与预警，其中有些数据只能在政府内部共享，不能被服务中介所利用。而有一些基础数据

或者具有结论性的数据可以对外公布，主要是对失信者实行惩戒，例如信用公示、警告、行政处罚、取消市场准入资格等，应建立由政府部门、授信管理机构、公共服务机构共同参加的联控联防机制。

目前，我国尚未建立完整统一的行政管理征信体系。主要政府职能部门、地方政府，特别是与经济活动相关的政府职能部门，都已经建立了自己的行政监管征信数据库，数据多少、质量优劣等各不相同，但都在对自己的行政管理职能发挥一定程度的作用。在社会实践中，这些政府职能部门的数据互联互通，实现统一的失信惩戒、守信奖励是非常必要的。未来一定会在国家层面新成立一个主管机构或由一个政府职能部门出面牵头，建立能够互联互通的国家级行政管理征信体系。特别值得注意的是，国家级行政管理征信体系建设的主要目的是要把各政府职能部门、地方政府的数据库联起来，实现信用信息共享。能够达到这个目的的手段有很多种，其中最经济、最有效率的就是建立信用信息交换平台，而不是建立大型的、全国的、实体的综合数据库。

3. 商业征信

商业征信是以行业协会组织及其会员为主导进行建设，以政府、企业、个人为主要用户，以企业和个人为征信对象，以信用信息在组织内部及相应市场范围内互联互通、共同防范信用交易与管理风险为主要目的的商业征信系统及信用管理运行机制的总称。商业征信体系的最大特征是市场化，它由独立于政府之外的民营机构构成并按市场方式运作。其信息特点是来源较为广泛，一般是交易性交换或有偿性提供的。商业征信体系主要由征信机构、信用评级机构等信用服务中介机构对企业的信用信息进行采集、筛选和评估等。

企业的信用信息包括企业市场交易的信息，也包括企业在商业银行贷款的基本记录，这些信息主要用于企业之间的交易活动。未来的征信服务行业既可以有公共征信机构也可以有私营征信机构，具体有做企业征信的，也有做个人征信的，但不论做什么，都应遵守市场原则，公平竞争，优胜劣汰。征信服务中介机构是市场经济发展、社会分工深化的产物，是征信体系的有效组成部分。它们按照现代企业制度建立，依据市场化原则运作，以独立、客观、公正的原则为市场提供产品与服务，为企业授信、雇主用工、投资合作、贸易融资等商业活动提供决策信息，是市场所需要的新兴现代服务业。

我国商业征信体系的建设和发展将取决于我国经济体制改革与深化的发展程度，即取决于市场化水平，具体地将取决于三大因素：一是授信机构是否真正承担风险、享受收益。这是我国商业征信体系生存的前提。在成熟的市场经济中，授信机构必须是独立的经济主体，必须按市场规则运作，独立承担风险并获取收益。二是受信人守信有益、失信受罚。这是我国商业征信体系发展的保障。只有这样才能做到赏罚分明，坚持正确的价值导向，保证市场交易与社会管理进入到一个良性循环。三是中介机构以信息商品与服务为业务能维持经营，其收益的大小取决于信用交易商品提供与服务需求的活跃程度。这是我国商业征信体系茁壮成长的条件。

9.5.4 应用价值提炼

随着互联网和大数据信息技术的发展,第三方机构的征信数据范围也在不断扩大和丰富,其应用价值也越发凸显:

- 更多更全面的维度来评估个人及企业的信用等级,降低各项业务的风险,增强信用信息的透明度,全面及时地掌握客户、供应商、合作企业的风险状况;
- 为企业商务交易及信用管理决策提供信息和评估支持;
- 为金融机构等主体与企业间的合作提供资信信息支持;
- 降低商务和信贷交易成本;促进企业信用记录、监督和约束机制的建立等。

9.5.5 场景总结回顾

我国征信伴随着金融保险业的发展已久,先后经历了四个阶段,分别是20世纪80年代的探索阶段、1996~2002年的起步阶段、2003~2013年的发展阶段和2013年至今的扩张阶段,但整体的征信体系较国外发达国家还是比较落后的。在此期间,我们还存在以下问题和风险:

第一,信用信息封锁。尤其是企业征信的信息,主要来源就是政府权威部门的发布、企业自愿提供、企业产业链条中的相关主体提供和新闻媒体介绍等。但是,由于现阶段国家立法对于征信公司合法使用政府信息并未有明确的法律规定,而依据现行法律征信公司,若想得到这些有效数据,其成本会极大增加。

第二,行业自律不足,监管机制缺位。发达国家的企业征信业均成立了行业协会。目前我国征信行业上没有行业协会,只是在上海等地成立了一些地方性的信用服务协会。整个中国征信业由于没有建立协会,所以行业内的交流、人员的教育培训以及制定行业执行技术标准(例如征信报告格式标准、数据库建立标准等)和执业规范,保障整个行业的利益等都不能提上日程,严重制约了企业征信行业的发展。

第三,机构规模小,从业人员素质参差不齐。尽管我国征信业在改革开放过程中得到了一定的经营规模,但与国外一些知名的征信企业比较,仍然显得势单力薄,规模偏小。在征信公司的品牌塑造中,起到灵魂主导作用的是征信企业的从业人员,这是因为,征信行业是建立在专业调查、精确细分的基础上的信息行业,因此人才培养极大地制约了行业规范化和标准化的过程。

第四,执业技术规范不统一。完备的法律法规和行业标准,是征信行业健康发展的保障。现代征信业身处信息化时代,信息技术发展日新月异,只有建立统一的行业标准,征信行业才能有公平竞争的环境,社会公众才能对各个不同的征信企业进行正确比较,有利于整个行业的规范运作和健康发展。

因此,大数据虽然在征信行业有了一定的应用,但还是会受到上述四个问题的困扰,这就导致了通过大数据获取的征信信息是否完全可靠的问题,也是因为这个结果,目前通过大

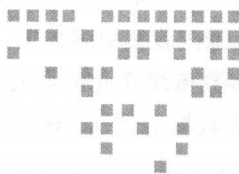
数据实现的个人和企业信用信息还更多地停留在简单的信息核对，很少机构能够给出一套完整规范的评分体系和评分标准，因为本身这套规范和标准在中国还是缺少行业协会指引的，但伴随着大数据和互联网技术的快速发展，我们有理由相信，很快就会出现类似的机构帮助建立完善的征信体系。

9.6 本章小结

大数据落地应用才是大数据发挥价值的最终落脚点，本章仅抛砖引玉，通过之前的部分案例讲解大数据业务应用的基本流程和思路，希望能给读者以启发。由于本章内容都是根据具体业务场景而实现的，因此很多具体执行方法可能无法具有通用性，但执行的逻辑、思考的方法、流程和主要场景等均可参考。

本章需要读者重点掌握的知识点包括：

- 大数据业务应用场景举例；
- 每个场景的主要应用流程、场景和价值点。



企业大数据价值评估

大数据对于企业的价值无疑是巨大的，从企业顶层战略、战术的制定到底层执行和实践，从资产端的价值变现到业务端的价值优化，它正在驱动企业运营策略、模式、方法的演进和变革。本章将介绍有关企业大数据价值评估的内容，重点从资产价值和业务价值两方面入手。

10.1 资产价值

大数据目前还没有被企业作为资产进行运营，但大数据具备作为企业资产的能力，原因是从资产的定义上看，大数据完全符合归属企业所有、基于过去的行为产生的现实资产、可以为企业带来经济利益的三个特征。从企业的资产视角看，大数据资产主要指的是大数据本身，即通过数据的本身流通可以进行数据变现和转化为企业经济收益。

大数据资产价值评估需要综合考虑数据规模、数据价值度、数据鲜活性、数据关联维度和数据粒度五个方面。

10.1.1 数据规模

数据规模可以简单地理解为数据量大小。对数据规模的评估可分为三个角度：数据存储容量、数据记录数和数据特征数。

数据存储容量

数据存储容量指的是数据占用的空间容量。随着数据源的不断扩充，数据存储容量已经不再是基于 TB 的存储，而是上升到 PB、EB、ZB 甚至更高层次。企业的数据规模主要受限

于业务体的自身规模、数据产生场景和业务属性，除了通过外部渠道获取外，企业通常只能在其自身可控范围内收集更多数据。

比如，大多数传统产业公司的数据大多以 MB 存储，较多的数据可能到达 GB 级；普通的电商类网站的数据能到达 TB 甚至 PB 的级别；而京东和阿里的数据能到达 PB、EB 甚至更高。这些通常都是由于企业规模的约束而导致数据容量的差异。而类似于 YouTube 这种视频类网站，由于视频业务的属性特征明显，其自身的数据容量会比文本、图片等更大，因此也往往具有较大的存储容量。

数据记录数

数据记录数通常是针对数据库型存储的数据而言，而使用非数据库存储的数据则没有“记录数”的概念，此时我们更多地会称之为数据样本量，即非结构化对象的存储数量。

单纯的数据存储容量大，并不意味着具备大量可用的数据样本。以下几种情况数据存储容量较大，但是却很难具备较高的数据价值：

- ❑ 单个数据对象大，但数据记录数较少。比如，一个视频文件可能达到 2GB，但实际上样本量只是一个，因此无法从中提取有效且稳定的知识规律。
- ❑ 数据库中的记录数少，但是数据维度非常多。比如，表中有 100 个客户，每个客户又采集到 1 万个维度，那么虽然整体容量也较为可观，但也无法提取有效数据规律，并且在海量维度下会存在数据稀疏、维度共线性等问题导致对数据结果的影响。
- ❑ 数据库中的记录数少，同时维度也不多，但每个维度的容量较大。这种情况主要集中在有大量的文本字段的场景下，由于大量文本的存在导致的数据容量“虚高”，该类数据也很难发挥较大价值。

数据特征数

数据特征指的是数据具备的可用维度，或者数据列。通常而言，更多的数据维度和特征更利于从多个角度进行挖掘和分析，并且有利于建立完整的数据认知。

比如，从用户的角度看，用户的特征包括社会属性、行为属性、兴趣属性、社交属性、学习属性、工作属性等，其中每个属性又可以再不断细分，这些属性对应到底层都是通过一个个的细分特征组合或计算生成的用户标签，由此产生完整用户画像。



数据特征数多并不意味着每次都需要把所有特征加入到模型中进行训练，这只是提供的可以从多个角度进行挖掘的条件，真正进行到模型中的特征必然是具有典型特征和代表意义的，而非全部。

10.1.2 数据价值度

大数据的一个典型特征是数据价值密度低，因此只有高价值的数据才能产生商业价值，也才能成为资产。除了价值密度低以外，大量的半结构化和非结构化数据由于受到现有技术手段和应用场景的限制，很难涌中挖掘巨大价值。数据价值会从实名、相对稳定、结构化、

业务相关、可验证等方面进行评估。

实名

实名指的是被标记的数据主体能通过实名信息进行查询和定位，实名相对的概念是虚拟。以用户在网络上的行为数据举例，大多数情况下，用户可能无需进行身份或实名验证便能使用互联网服务，此时采集到的数据都是基于 Cookie（或其他技术方式）的匿名用户，无法对应到真实用户个体。这些数据的应用场景只能是互联网的部分应用，例如广告投放、个性化推荐、用户兴趣等，无法跟其他数据做打通应用。



提示 在互联网上的数据主流的识别方式都是基于 Cookie 进行的，无论是来源渠道、浏览行为、订单转化等都通过 Cookie 进行匹配，只有当用户具备实名条件才能与真实用户信息相关联，例如实名认证、上传护照等。这些数据相对于实名用户数据虽然规模大，但其价值点相对较小。

相对稳定

相对稳定的数据是数据产生价值的基础，如果数据变动性太多或无规律性，那么数据即使获取到也可能因为“过期”而无法使用，相对稳定与更新频率有关。比如，订单状态的数据通常取决于物流配送的周期，从下单、审核、支付、发货、收货、确认到退换货的整个过程都会影响订单状态。如果看整个数据，订单状态会在整个订单周期内发生变更，因此是不够稳定的。而这些数据如果不能进行及时更新，那么这些数据的可信度较低。

结构化（或者可转换为结构化）

由于基于结构化的数据处理方式相对成熟，因此在当前技术方式的限制下，结构化的数据更利于数据挖掘和机器学习，而半结构化和非结构化数据价值提炼难度较大，甚至很多非结构化的数据无法统计数据规律。以手写识别为例（这是一个典型的针对非结构化数据的监督式机器学习模式），通常需要先先将图片转换为一个结构化的特征矩阵，然后再基于特征矩阵和预置标签进行模型训练，最终对新输入的手写图像进行识别（大多数是一个分类的过程）。当然，随着科学技术的进一步发展，很多非结构化的数据也会具备直接处理的能力，而无需预先转换为结构化数据。

业务相关

由于不同企业、不同业务对数据的需求不同，因此数据价值的高低与业务需求关联性有关。比如，金融类行业对交易类、客户类、企业类数据更感兴趣；社交类企业更能从用户行为、网络声音中提取商业价值；制造类企业对于客户需求、商品原材料、物流配送等更看重。



提示 业务相关还意味着与核心业务最相关的数据更具有价值，例如销售类公司对于销售数据更感兴趣，因为可以直接从销售数据中分析商品进销存特性，而对于更靠前端或离销售环节更远的客流引入、营销推广的兴趣度则会小一些。

可验证

可验证性要求数据的各个维度和数据质量是可以通过一定途径进行验证的。但当前大多数数据都是很难验证，尤其是大量的复合数据、汇总数据、匿名数据导致的数据无法追踪和还原问题更加严重。数据可验证意味着数据的真实性、有效性、统一性、及时性、准确性可以进行验证，这是数据产生价值的基本条件。



提示 数据不可验证的另外一个关键因素是国家缺少统一的法律和法规保护，导致各个行业和企业间进行数据交互时，基于自身企业数据安全的考虑而对关键字段进行加密，导致输出的数据是匿名的。

10.1.3 数据鲜活性

在数据价值中，具备高价值的数据需要相对稳定的特征，客观上要求数据是“鲜活”的。这意味着数据需要及时更新以便获得更新、更准确的数据。数据鲜活性对数据有两方面要求：一是数据本身的鲜活性；二是数据更新的及时性。

数据本身的鲜活性意味着，数据在采集端是及时更新的，表现为数据会随着业务或数据主体的变化而及时更新。比如，当用户的学历已经从本科变更为研究生时，数据也要随之发生改变。

数据更新的鲜活性指的是数据需要及时更新和同步，这是数据更新的周期和频率，例如毫秒级、分钟级、小时级、天级等。通常，数据越实时，理论上数据的鲜活性会越高。



提示 数据本身的鲜活性跟数据更新的鲜活性有关，如果数据本身不容易发生改变，那么即使是久远的数据也一样可以保持“鲜活”。比如，性别字段通常无需更新；而收入指标在不同时间内通常会产生波动，要想获得更准确的数据，1个月前的收入将比1年前的收入更可信；对于用户年龄，在1年内的数据都无需更新。

10.1.4 数据关联维度

基于用户碎片化行为、产业链服务和社会化分工的背景，所有拥有数据的企业均只具备各自领域内的部分数据。要实现完整的数据洞察和认知，需要将所有相关的数据维度关联并整合起来。要实现数据整合，要求数据必须具备可关联条件或可关联属性。

对于数据而言，可供关联的维度越多，则越能将多种场景进行整合。以用户的数据为例：

- ❑ 通过身份证号可以关联到用户的基本属性信息，包括性别、年龄、生日等；
- ❑ 通过订单 ID 可以关联到用户的购物信息，了解用户的购物习惯；
- ❑ 通过商品 ID 可以关联到用户喜爱的商品特征、属性、品类、价格等；
- ❑ 通过 Cookie 信息可以关联到用户在互联网上的行为信息，了解用户网络行为喜好；
- ❑ 通过社交 ID 可以关联到用户的社交行为和属性，深入了解用户社交数据；

除了上述可关联场景外,还有更多基于用户的场景,所有场景下各自具有多种关联维度,并且只能在对应场景下应用,例如购物中的订单 ID 和商品 ID、社交行为中的社交 ID、网络行为中的 CookieID 等,具备更多的维度才能将更多的信息关联。

除了可以关联的维度更多外,还要求这些关联维度是唯一的、通用的、可识别、不变的、健全的:

- 唯一的:不同数据主体间是可区分的,用户的姓名由于会重复,因此不能作为唯一用户关联项。
- 通用的:关联标志能被各个场景识别并能成为统一标准,而不应该只针对特定场景才能识别,例如汉字虽然可以作为关联项,但在某些环境下可能产生乱码,因此无法成为通用的关联标准,而阿拉伯数字和英文字符则可以。
- 可识别:通过标志可对应到特定的数据主体,例如通过学号可以对应到不同的学生,但如果没有学生的进一步信息,则无法识别出该学生的具体信息。
- 不变的:识别标志不能发生变化,否则将失去对主体的标识,例如手机号通常在一定周期内保持不变,但拉长时间来看,用户很有可能更换手机号,因此手机号不是用户不变的标志。
- 健全的:这要求数据的可关联维度必须对所有数据记录都存在,而不能存在某些场景下无法获取,这样会导致关联无效。

10.1.5 数据粒度

细粒度的数据是数据整合的基础,特定粒度的数据也是进行数据计算和应用的前提条件。不同粒度的数据其价值有所差异,数据粒度越细,数据的可挖掘价值越大。同一种数据主体对象的数据粒度的评估根据不同数据业务场景和价值而有所差异。以商品数据举例,在订单时最细粒度是订单商品,在进销时最细粒度是每次商品(货物)进出记录。

通常而言,基于每个事件的数据可利用价值最高;其次是基于各个层级维度的汇总统计;最后是整体的汇总统计。比如,用户网络行为日志基于每次请求产生,因此可以基于用户个体进行分析,并可以产生基于每次请求的个性化推荐;某些西部地区电力部门通常每日或每小时更新一次数据,只能基于天或小时来提供电力应用场景。

10.2 业务价值

大数据在企业运营过程中扮演着非常重要的角色。在业务价值上,大数据价值主要体现在用户体验提升、运营优化、销售贡献、供应链优化等方面。

10.2.1 用户体验提升

大数据应用于用户体验是多数企业相对成熟的价值落地点,主要价值场景包括:

- 通过数据分析用户需求，并个性化地匹配商品、资讯或服务等项目；
- 通过对用户生命周期的全局性了解，发现不同用户在生命周期内的活跃、价值情况，并针对性地开展用户发掘、激活、挽回、沉淀等模式，提升用户生命周期价值和体验价值；
- 通过对用户全数据的整合，覆盖用户属性、行为、兴趣、社交、购物、学习、工作、家庭等方面，从而描绘出用户 360° 的特征标签，为用户提供最优的行为路线、模式引导、产品应用、构造服务等；
- 通过关联外部数据，全面掌握用户的“隐藏”信息，从而促进建立更加有效的与用户交互、沟通的渠道，在任何时间、地点下都能保证用户的需求可获取、可触达、可沟通、可满足。

对于大数据在用户体验方面的价值评估，主要侧重于企业维度和感知维度。

企业维度

企业维度指的是用户体验对企业本身的反应，它是用户体验在整体上对企业或品牌的认知，包括联想度、满意度、信任度、吸引度等。

- 联想度。联想度指的是当用户具有某些需求或触发某种场景时，可以在第一时间联想到企业提供的相关服务或商品。被联想到的主体可能包括产品属性、使用场景、消费者利益体现、品牌口号等。比如，当我们想要购买大屏手机时，第一时间可能会联想到苹果 iPhone6 Plus。
- 满意度。满意度是客户对产品或服务的满意程度，它是基于客户预期与实际体验的相对关系而产生。客户的满意度受到主客观很多因素的影响，客户主观上对企业各个方面的认知以及客观上企业自身在与客户的所有接触点上的实际体验差异都会影响满意度。比如，我们在网上购买商品时的物流配送普遍是 2 天到货，而某些企业可以当天下单当天送达，那么在配送满意度上就会大幅度提升。
- 信任度。信任度是客户对企业信任的程度，它基于客户对企业信息掌握的基础上，结合家庭背景、文化程度、生活阅历、接触时间、了解程度等因素产生。比如，在 10 年前大家都很难接受预先给网络平台支付购物费用，但现在网络购物预付款的方式已经可以被大多数网络消费者认可。
- 吸引度。吸引度是企业对客户的吸引程度，表现在实际中是客户围绕企业或品牌形成的社交圈或粉丝圈。比如，苹果、小米、华为等各自都有忠诚客户，并且形成了粉丝团。粉丝团中不乏意见领袖，他们对于企业口碑的传播和品牌的建立具有重要作用。

感知维度

感知维度泛指企业为客户提供的可以实际触达并感知到的接触点，例如商品、购物平台、配送服务、客服中心等，通过这些接触点为客户带来的用户体验上的直接反馈和感知，包括活跃度、新鲜度、易用性、价值度等。

- 活跃度。活跃度指的是用户活跃的程度，可以从访问、注册、登录、搜索、点击、购

物、评论、咨询、发帖等角度进行评估,评估的维度包括数量、时间、金额、进入、退出、转化等。活跃度可以基于单一指标,也可以通过对多维度的综合计算得到。

- 新鲜度。这里的新鲜度不是指用户的新老程度,而是用户对于企业各种触点的新鲜感知。在一定周期内,用户希望获得稳定且统一的触发方式和应用;但同一种触达在经过一定周期之后会导致用户接受疲劳,导致用户体验的下降。比如,电商网站策划各种主题的促销活动、网站页面不断改版、美食中心不断更新菜谱都是提升用户新鲜度的方式。
- 易用性。易用性指的是企业提供的产品、服务、平台等容易被用户使用,无需用户具备特定的知识背景并且不需要进行复杂的知识和模式的学习便能掌握应用方法。易用性的前提是可用,可用意味着产品、服务、平台等能够发挥作用,而易用则体现了使用过程中的便利性和简易性。
- 价值度。价值度指的是用户对企业发展的价值贡献程度,所有用户体验的最终落脚点都应该转化为企业发展的助力。但并不是所有的价值都以付费贡献作为参照,价值贡献根据企业不同的发展阶段和模式而定。比如,对免费工具而言,客户数量和活跃程度是价值贡献的标准;对导购类网站来讲,成功的提供对第三方购物线索的引入是价值贡献的标准(订单本身在第三方网站产生,而非导购网站本身);对广告类企业来讲,广告曝光、点击以及特定事件则是广告效果价值的关键。

10.2.2 运营优化

大数据对于企业运营管理的优化可以覆盖全企业运营过程,从每个业务环节的执行流程上看,覆盖决策、执行、评估三个环节。

决策制定

在制定决策时,大数据可以基于历史信息形成对未来的预估和推断,以提供辅助决策信息。它可以帮助企业建立合理的预期与目标,并为实现目标建立资源需求图谱和协调方案;同时,还能够帮助企业提前识别未来会发生的异常情况,通过建立相关机制减少或避免损失。常见的价值场景包括:

- 未来一个月内,商品价格会下降 20%,针对性的商品价格保护与增值服务策略需要调整。
- 本次促销活动预计响应率会达到 80%,带来 300 万元的订单收入,巨大订单带来的商品需求量提高预示着从商品制造、库存、物流配送、平台销售、客户支持的整体体系都要进行扩能。
- 预计下周会员活跃度会从一般活跃上升到非常活跃,针对会员的管理和营销、支持、服务部门在这个过程中的效果较好,可以针对性地提高公司资源的倾斜。
- 公司计划下个月网站日均流量会在 300 万~500 万,网站响应和支持能力、营销费用、购物流程等的合理保障是将流量转化为销售的关键。

业务执行

在业务执行时，除了需要沿着企业整体布局与战略的方向进行贯彻落实外，更重要的是需要明确如何科学、正确并有效地实施战术落地。大数据可提供数据挖掘的结果为业务使用，并能进行规则引导。这类场景常见于业务有明确的行动目标，但需要找到具备一定特征的数据要素作为业务执行的参照，根据数据提供的内容是否明确，大数据的价值落地场景包括两类：

第一类是具有明确的业务执行规则，数据规则可直接被业务使用，价值场景如下：

- ❑ 现要针对网站预计会流失的会员做会员挽回，应该挑选具有什么特征的会员？——收入 > 5400 元，最近购买时间是 5 个月之前，总订单金额在 4300 元以下的会员。
- ❑ 商品 A 库存大量积压，现要将该商品进行捆绑和搭配销售，应该选择哪些商品作为捆绑对象？——与 A 商品关联销售规则较强的商品是 C/E/G 商品，这些商品搭配销售预期提升 300 万元的订单收入。
- ❑ 网站需要新增广告位以满足越来越多的商家广告需求，应该在哪些位置新增广告位？——首页右侧区域，用户点击率较高，该位置可考虑开辟为新的广告位。

以上规则明确了业务所要行动的细节要素，是一种具有极高落地价值的数据工作。

第二类是提供模糊的业务执行规则，提供的是执行方向和路径，价值场景包括：

- ❑ 某商品 E 页面流量来源中，站内流量来源太少，现要提高站内流量入口，如何实现？——站内主要流量页面是 A/B/C，可以从 A/B/C 三个最大流量的页面入手。
- ❑ 今日大型促销活动下，不少线下商贩也加入到普通消费者队伍中抢购商品，这些贩子都是哪些人？——根据数据挖掘结果提供了类似贩子的异常会员 ID，需要业务方进一步核实。

这些规则是给业务人员相对明确的方向引导，是业务有效完成工作的指示灯。

效果评估

大数据在企业各个运营角度的效果评估的价值是应用最多的场景，数据在效果评估方向的基本落地地点是对业务效果进行理性定义以及针对结论的深入原因分析和挖掘。

(1) 效果定义

效果定义是对正在发生的现在和已经发生的过去做出结果判断，以评估结果是否符合预期或存在异常情况。结论定义并不是简单地定义结果是好还是不好，而是要进一步定义所谓的好或者不好属于正常还是异常情况，这才是真正的数据结论定义。现在很多数据分析师在给出结论时往往是这样的陈述“昨日比前日增长 20%”“流量下降 40 万”，类似这样的报告不属于结论定义，这只是数据陈述而已。

结论定义最常应用的场景是业务状态进行时和业务状态完成后。业务状态进行时的结论定义可快速帮助业务建立实时数据反馈机制，通过即时的数据结果判断是否符合预期，并可通过措施优化当前业务状态；业务状态完成后的结论定义除了可以做业务效果评估外，还为原因解析和数据探究提供了方向。

常见的结果定义场景：

- 昨日订单量超过 30000 单，超过正常水平 230%。
- 过去的 1 小时内流量突然下降了 75%，这是一个异常的预警信号。
- 过去一周内的注册会员量环比增长 7%，这是正常波动。
- 晚上 6 点流量下降到 50 万在线 UV，这是正常流量下降。

（2）效果分析

效果分析是对数据进行探索和研究以便发现进一步的数据观点和数据洞察，这是一个数据探究的过程。效果分析是挖掘数据深层次原因和关系的关键动作，也是数据论证的主要过程，表现在数据结果中大多是数据论证过程。

在效果分析时的主要场景是针对已经明确或知晓的结论的分析。这是围绕已知结论进行数据分析和挖掘，以找到导致结果发生的原因。在业务应用中，常见场景是针对业务提出的具体问题分析，侧重于“为什么”的答疑解惑，例如：

- 昨日公司销售额提高 77%，是哪些原因导致的？
- 最近一周公司日均注册量下降 7899，是什么原因导致注册下降如此严重？
- 最近网站订单转化率提升 15% 等，是由于购物车、流量提高还是站内活动等因素导致？

还有一类分析是没有明确的数据结论，只围绕某一范围或主题开展数据挖掘工作，以便寻找一定结论和原因的过程。

针对未知结论的数据探究是拓展业务知识的重要途径，相比较针对已知结论的数据探究，该过程更侧重于“是什么”的工作范畴，常见场景是：

- 不同的商品是如何关联销售的？
- 企业整体用户特征是怎样的？
- 页面商品布局中，哪些因素会提高页面点击购买转化率？

在针对运营优化的过程中，主要评估的方向包括以下几个方面：

- 效率性。高效的运营会提高企业的精细化和科学化运作能力，在资产和费用投入不变的前提下提高经营产出，以提高单位主体的产出。对于动作的平均链条长度、执行时间等都是效率的主要评估维度。
- 及时性。无论是战略战术的策略执行还是业务执行，快速响应能力有利于企业应对复杂多变的商业环境与竞争压力，借助于实时计算和流式计算，可以缩短决策周期并提高决策的时效性。对及时性的评估更侧重于对出现事件时的相应时间，例如分钟、小时等级别。
- 准确性。基于历史海量数据，可以产生对决策和执行的预测结果，通过预测结果自身的准确性数据（包括准确率、提升度、响应度等），以及与实际执行结果的比对，可以找到数据决策的准确性。所有基于预测的模型都可以提供预测准确率的数据。
- 风险性。在使用大数据从海量数据中获得决策辅助支持时，可以通过数据对业务中的不确定性进行分析，以得到难以通过业务经验发掘的异常因素；再结合业务经验，通

常可以提高对运营风险性的评估和应对。在风险评估时的数据规则可信度、业务周期、预测输出值等都是评估维度。

- ❑ 可行性。可行性指的是决策是否具备执行的客观条件。在决策时可以通过在大数据虚拟业务系统中进行预先计算，来判断决策依赖条件、影响范围、影响周期、是否可控等因素，以验证决策的可行性。

10.2.3 销售贡献

大数据应用于商品和业务销售方面，可以提升销售额和利润，具体价值场景如下：

- ❑ 通过挖掘出不同商品和服务间的关联销售模式，对商品进行打包整合销售以提高客单价。
- ❑ 通过向上销售找到客户在不同购物模式之间，以及前后相关联的销售规律，对客户进行复购引导，提高复购率。
- ❑ 通过对商品和服务的价格敏感度的分析，针对不同人群实行不同的价格策略，再结合不同的促销方式、优惠券、特定活动、会员等级等进行精准销售，提升综合利润水平。
- ❑ 通过商品和服务的画像分析，得到不同人群对应的不同商品或服务的偏好，然后针对不同人群提供不同的商品或服务，以提升商品或服务的转化率。
- ❑ 通过对商品生命周期的了解，掌握不同商品在不同周期内的销售表现，然后针对性地制定销售策略，以实现良好的商品周转并减少库存积压。

对于销售贡献的评估，主要侧重于两个维度：一是结论维度；二是过程维度。

结论维度包括直接销售收入 / 利润和间接销售收入 / 利润，带来销售的提升是销售贡献的终极目标。

- ❑ 直接销售收入或利润指的是通过数据方式直接促进商品的销售而带来的收入或利润，例如通过挖掘用户需求后针对不同人群推送的商品或服务，并被客户接收的部分。这是数据产生销售贡献的直接方式。
- ❑ 间接销售收入或利润指的是数据在发挥作用的过程中，是与业务一起协同开展的，表现在实际场景中是业务同时还执行了其他方面的优化，然后再借助于数据提供的建议或优化方式协同执行，最终促进销售收入或利润的提升。这种情况下，需要对比在数据介入前和介入后，在保持其他业务执行要素不变的情况下，数据给业务带来的提升。



提示 实际上，在不同场景下，很难控制客观上的所有要素都保持不变或完全相同，即使能控制所有的内部运营要素，外部客户需求、竞争对手、行业变化等都无法控制完全不变，因此这种对比方法只是相对的，并且具备一定的（具体程度要看可控制的要素）参考价值而非 100% 准确的。

过程维度指的是在对产品销售提升的中间过程中的相关维度，包括：

- ❑ 复购率：复购率指的是同一件商品被购买 2 次或 2 次以上的用户占总体用户的比例。

不同公司对于复购的定义也可能增加时间的维度,复购意味着用户能够重复消费。

□ 转化率:转化率指的是完成转化的用户占总体用户的比例,转化可以被定义为任何与销售相关的中间过程,例如加入购物车、结算、提交订单、支付成功等。

□ 客/订单/件单价:客/件单价指的是每个客户、商品或订单的平均价格,在数量一定的情况下,单价越高,销售收入越高。

□ 平均订单/商品数量:平均订单/商品数量指的是平均每个订单内的商品数量,或购买的每种商品的数量。在订单数量一定的前提下,订单内的商品数量越多,销售越高。

除了这些与商品销售直接相关的评估维度外,还可能包括网站流量数量类指标、网站流量质量类指标、客户服务类指标等,这些都与销售贡献相关,在此不作展开。

10.2.4 供应链优化

在产业协同的背景下,供应链优化是大数据发挥作用的重要价值方向,它的基础是供应链上各个环节企业的协同工作,并能实现多方数据互通。基于大数据的供应链优化价值有利于解决以下问题:

- 如何在保证产品质量的前提下,降低采购原料的成本?
- 如何通过数据更好地预测商品采购量,并降低库存积压成本?
- 如何合理分配商品在不同地域间的库存分配,降低物流和配送成本?
- 如何通过与供应链上下游的协同,降低市场和竞争风险?
- 如何通过供应链一体化提高行业进入壁垒,保证企业在市场竞争中的优势地位?
- 当前市场的需求表现在哪些方面?如何在商品和服务中满足市场需求?
- 如何利用上游市场供应与下游市场需求信息,提高库存周转?

大数据在针对供应链的优化过程中,主要评估以下几方面:

- 原料获取。通过更加全面和透明的信息,进行原料采购、谈判与合作经营,从而为新产品和服务找到质优价廉的原材料,原料的单位成本、总体费用、材料规模、供应效率、供应质量、流程标准化等是评估的重点。
- 采购预测。需求是采购的基础,基于大数据可以对需求进行预测,从而产生更加精准的采购需求。基于精准需求的采购更能协调内部进、销、存的各个环节,防止脱销和滞销,因此采购的重点评估是针对采购成本、规模、结构、质量、效率、时间等。
- 库存优化。商品库存一直是销售类企业资金占用的重要组成,通过大数据实现对安全库存的管理,提高库存周转,因此对于周转成本、周转效率、库龄、库存结构等方面的评估是主要方向。
- 反向研发。通过对市场和客户需求的数据洞察,反向促进商品生产和制造的研发和计划,形成以需定产的良好模式,增强商品与市场需求的匹配度,评估时的重点是市场销售的规模和利润结果。
- 物流配送。基于大数据建立智能调控和物流配送控制体系,可以结合前端生产和后端

需求，合理制订不同仓储的物流配送计划，并建立动态管控管理模式，将供需的传送效率科学化、合理化、最优化，评估过程中的满足率、满意度、及时性、有效率、成本效率等是主要因素。

- 风险预测。基于完整供应链的大数据，可以精准把握市场动态、客户需求变化等市场风险因素，提高供应链整体的应对能力并建立针对性的应对机制；同时，基于大数据可以建立供应链的问题追踪机制，保证在出现问题时可采集、可跟踪、可反馈、可解决。

10.3 本章小结

大数据能够帮助企业将数据资产变现，并大幅提升企业的商业决策水平，降低企业经营的风险。大数据在企业中不仅是辅助决策的角色，更会在企业商业决策和商业价值的决策中形成从辅助到驱动的角色转变。

本章需要读者重点掌握的知识点包括：

- 大数据作为企业资产的价值及评估方向；
- 大数据对于业务价值提升的四个关键点。

大数据的社会价值

随着大数据观念、技术和应用的不断普及,基于大数据出现的很多应用场景和业务模式已经开始颠覆我们的传统想象和认知。大数据不仅对微观企业具有巨大的推动作用,还在民生、政务和产业等宏观领域意义极大,正成为推动社会变革和经济发展的强大动力。

大数据对民生、政务和产业到底能带来哪些变革和价值驱动?本章将试图探索这些问题。

11.1 民生价值

面向民生领域的大数据开放和应用是大数据产业发展的核心价值之一。基于大数据的智慧民生将主要体现在医疗卫生、交通出行、家庭生活、就业工作、教育学习、社会保障等方面。

(1) 医疗卫生

通过建立统一、共享的医疗卫生数据系统,将个人的电子档案和病例、医疗和社会机构的卫生服务与供应管理数据相结合,为医疗业务、卫生活动、健康体检等服务提供基础。医疗卫生数据系统可以整合医疗卫生数据中大量医疗体系的数据,例如非结构化和半结构化的病例、文档、图像、视频等,个人的各种行为和属性数据如可穿戴设备、健康记录、运动膳食、用药护理等信息,可应用于三方面:一是建立个人全生命周期的医疗管理模型和健康医疗解决方案,涵盖疾病预测和防护、病情精准诊断、病理原因探究、精准用药和个性化治疗、远程监管和看护、健康生活管理等个人健康生活的全流程。二是协助建立社会保障性医疗调控体系,涵盖医疗资源调控、关联资源配置、重大疾病探测、传播疫情控制、紧急事件应对等完整知识体系和处理机制。三是有利于促进医药行业的研究和开发革新,有利于缩短医疗研发周期、精准选择试验用药人员、新药品效果跟踪、个性化基因研究等。

（2）交通出行

交通是资源流通的动脉，更是经济发展的基础要素。大型城市中普遍存在交通问题，例如交通拥堵、交通环境污染、交通事故、交通违章等，借助于大数据技术，智能交通管理和智能出行应运而生。大数据在智能交通管理和智能出行上的作用主要体现在五个方面：一是辅助城市规划，通过对交通出行的监控及城市配套和服务设置的分析，合理规划交通路网与功能区分布。二是辅助交通引导，智能交通可以实时监控车流量，基于对车流量的预估和判断，可以针对性地进行路线动态引导、公共交通调配、公共需求管理等。三是交通违法监管，智能交通管理可以对公共交通环境中的违法行为进行监控、分析、识别，大大降低人工参与监控的程度，有利于提高监管力度并净化交通风气。四是车辆辅助驾驶，车辆中通过对气候条件、道路环境、交通路网、车流量、车辆信息等的采集、分析和计算，辅助驾驶员出行，可以有效减少人为疏忽造成的交通事故，未来自动驾驶的出现将大大降低事故发生的概率，保障驾驶员生命和财产安全。五是交通衍生服务，利用交通出行信息的相关性，可以针对性地提供周边泊车、维修、代驾、违章查询、证照办理、加油站、各类公共交通时刻表查询等综合信息服务，为人们出行提供便利。

（3）家庭生活

家庭生活已经是大数据的主要应用领域。未来，我们的周围将出现各种有关智能生活的系统和产品，并且它们会在家庭生活中扮演着非常重要的角色，包括智能移动、智能社交、智能家居、智能购物、智能办公等方面。智能生活可以为人们提供智能安防、设备控制、环境监测、远程遥控、对象识别、趋势预测、程序执行等功能，用科技的方式来提升衣、食、住、行和工作、娱乐、生活、学习的安全性、便利性、舒适性。比如，在智能购物方面，大数据可以根据个人的消费记录来推荐内容，帮助人们选择最匹配的消费方式和消费内容，同时以最小的成本获得最大的生活品质提升；在智能家居方面，通过远程控制监控家庭是否遭到入侵和盗窃，然后可针对性地触发警报和安防系统，保护家庭人身和财产安全。

（4）就业工作

大数据是促进就业的重要方向，也是未来高科技人才的重要蓄水池。据美国劳工部预测，到2018年数据分析师的需求量将增长20%，由于中国大数据起步晚人才匮乏，因此人才缺口更大。当前大数据的人才主要集中在互联网等高科技领域，随着大数据落地步伐的加快，大数据需求和人才落地会迅速蔓延到其他领域及行业，其就业前景已经成为与银行、金融、消费品、咨询、地产等相当的热门方向，大数据分析师更是被媒体称为“未来最具发展潜力的职业之一”。大数据人才的需求大致有三个方向：一是技术类人才，包括技术开发、系统运维、软硬件维护、系统实施、虚拟化服务等；二是数据类人才，包括统计、数学、数据挖掘、数据算法、机器学习、自然语音处理等；三是业务类人才，专属于各行各业且具有一定业务经验的知识型人才。

（5）教育学习

互联网、大数据、云服务、物联网等技术和理念正在引发新一轮的知识沉淀、学习创新

与教育改革。在知识沉淀方面,大数据中蕴含的宝贵财富,在教育过程中经过不断总结、沉淀和积累,可以形成教育智库以供传承。在学习创新方面,大数据可以驱动以个人为中心(而非以老师为中心)的全生命周期学习模式的建立与完善,将家庭教育、学校教育、社会教育结合起来,促进终生教育和终生学习文化的形成。在教育改革方面,大数据可以为教育宏观决策提供数据基础,用于资源分配、人才规划、人才诊断等方面,让决策有数据可依;大数据可以推进学习和教育的个性化和差异化模式,借助数字化、网络化、共享化等方式推动学习、评估、引导、培训、激励的全面性、有效性和公平性,提升教育的综合质量;大数据可以提升教育和教学水平,基于数据驱动的教育改革和模式评估有章可循,大数据是推动发展智慧教育的基石。

(6) 社会保障

大数据是建立统筹城乡一体数据,涵盖社会救助、社会保险、社会福利、社会保障的社会保障体系的基础。它能有效解决统筹层次低、流程低效复杂、数据不同步、信息不互通、资源不共享等导致的政府保障资金不透明、企业偷漏缴纳保费、个人骗保重保和重复报销等问题。大数据在面临这些复杂社会问题和矛盾时提供了新的途径和思路,有助于建立前瞻性、准确性、公开性、互动性的社会保障体系。一方面,大数据可以为社保建立监管、稽查、评估和预警机制,可提升政府在社保资金的征缴、运营、发送、跟踪和处理的能力,保障公民实时了解自身权限;另一方面,基于大数据可以建立真实有效的动态人口管理体系,并对人口的现状、趋势等的研究和决策分析提供依据,可以推动社会管理创新和公共服务均衡发展,以实现社会保障管理效益的最大化。

另外,大数据还能推进民生业务的网络化和自动化,优化业务运转流程,为人民在证件办理、资质服务、经营纳税、婚育收养、公共事业等公共服务事项中提供便利的预约、查询、办理、跟踪、应用等内容,促进大数据技术成果惠及民众。

11.2 政务价值

大数据参与政务治理,是大数据时代的必然趋势,更是建设智慧政务的必要手段。大数据在政府对外事务处理和对内事务管理中扮演着越来越重要的作用。

1. 对外事务处理

在信息时代,大国之间的关系处理已经不限于物质层面,更涉及信息、数据等非物质层面。在外对事务处理过程中,大数据可以应用到提升综合国力、突破技术封锁、维护国家安全和促进国际合作等方面。

(1) 提升综合国力

当前,国家竞争的焦点已经从资本、土地、人口、资源的争夺转向对大数据资源的争夺。大数据资源已经成为国家核心战略资产,正在成为改变世界各国的综合国力,以及重塑未来国际的战略格局和势力范围的决定性因素。未来国家层面的数据国力博弈将体现为国家

所拥有的数据类型、规模、质量、价值等基础数据层面，核心技术专利层面以及数据挖掘和价值产出层面。数据主权将成为继领地、领海、领空之后的第四个国家主权要素。习近平在中央网络安全和信息化领导小组第一次会议上指出：“网络信息是跨国界流动的，信息流引领技术流、资金流、人才流，信息资源日益成为重要生产要素和社会财富，信息掌握的多寡成为国家软实力和竞争力的重要标志。”

（2）突破技术封锁

科学技术是第一生产力，科技强则国强。当今世界级的技术制高点主要由欧美发达国家掌控，中国由于大数据事业起步较晚，相关技术的核心竞争力仍然处于落后地位。这使得在大数据工作中发挥关键作用的核心技术将主要依赖于国际共享和合作，这种依赖关系可能会导致由于国外对核心技术的封锁使得大数据关键工作难以开展。这种技术封锁已经对实现国家大数据战略的落地形成巨大挑战，同时由于技术的落后也会对数据主权完整与国家信息安全造成威胁。因此，大数据相关的知识和技术创新、立法保护、资源积累和知识沉淀等非常重要。在国内巨大市场需求的推动及政策鼓励下，我国部分机构和企业已经具备了大数据相关技术的创新和研发能力，并且取得了一定的技术成果和产业应用。但是，知识和技术创新绝不是个别机构和企业的事情，而是国家每个企业和个人的事情，创新还需要全民参与。

（3）维护国家安全

国家安全包括政治安全、国土安全、军事安全、经济安全、文化安全、社会安全、科技安全、信息安全、生态安全、资源安全、核安全等。大数据可以增强国家不被外部威胁和侵害的能力。主要体现在三个方面：第一，大数据将有助于建立“观察-预测-决策-执行-评估”的整个决策和应对机制，将被动防御和作战转变为主动应对和作战，提升国家防卫的决策速度、反应速度和应对效果，促进基于数据驱动的现代决策系统的形成。第二，大数据可以整合所有类型和来源的信息，并能从海量数据中挖掘事物间的潜在规律，这能有效地将各种不利于国家安全的事件和因素扼杀在摇篮之中，做到预防性防卫。第三，由于中国具有丰富的大数据资源，并且大数据会源源不断地产生与利用，大数据资源永远不会枯竭，因此大数据可提供持续的保护国家安全的能力与价值。

（4）促进国际合作

未来，中国凭借自身丰富的数据和技术资源，可以广泛地参与国际分工与合作。通过坚持平等合作、互利共赢的原则，建立完善的大数据国际合作机制以及基于大数据衍生的政治、经济、文化、社会等方面的合作，充分利用国际创新资源形成与国内资源的互补，促进大数据相关产业、技术、知识和应用的发展，同时带动大数据生态的繁荣。对于国内有核心竞争力的大型骨干企业和集团，有计划、有步骤、有目标地引导与国际优势企业加强大数据关键技术和产品的交流、合作与研发；同时，支持国内大数据企业参与国际市场竞争，积极开拓海外业务，形成若干具有国际竞争力的大数据企业标杆。

2. 对内事务管理

以大数据为核心，依托完整、实时、开放、共享的大数据平台建立数据决策、数据管

理、数据评估的管理机制,有利于推动政府管理理念和社会治理模式的进步,在加快政府转型、提高行政效率、创新政府管理、促进服务升级、规范权力监督、维护社会稳定等方面具有重要意义。

(1) 加快政府转型

党的十八届三中全会描绘了全面深化改革的新蓝图,对加快转变政府职能、深化行政体制改革、建设法治政府和服务型政府提出了新的要求。加快政府职能转变,促进管理型政府向服务型政府的转变是深化行政体制改革的核心。大数据可以为政府转型提供决策参考,制定科学的转型设计方案与阶段性实施思路;同时,可以有效评估转型过程中,政府对于市场的干预情况,包括干预范围、程度、时间、周期、内容等;更重要的是,可以通过大数据评估和量化政府转型过程中对市场发展的具体影响,真正把转型落地和效果评估结合起来分析。

(2) 提高行政效率

管理效率和管理能力是提升政府治理能力的重要标准。管理能力离不开科学决策,在制定相关政策、法律、法规时,通过分析大数据得到的知识和规律,提升对国家政治、经济、社会文化、环境等方面的完整认知,依托客观规律及知识推演进行科学决策,以减少决策偏差对管理实施效果和效率的影响。另外,还可以针对落地过程中涉及的内容、环节、要素建立预演模型,通过预先设置模型中的条件变化来推导可能产生影响的因素、范围、程度等,通过科学化、数字化的组合测算来建立动态规划和决策机制。

(3) 创新政府管理

大数据是推动政府从“权威治理”到“数据质量”和“数据强国”的关键。政府管理围绕社会活动展开,通过对民生、产业、政务及客观环境的数据分析,可以有效获得管理范围内各个主体和对象的实际情况及真实动向。在政策范围内,基于已经掌握的社会诉求和对未来的预测,提供新型的政府管理模式。比如,通过智能交通系统来实现新型的交通管理,以达到交通分流和减少空气污染并节省出行时间;依托开放式的网络互动平台来增加政府与公众的沟通渠道,提升公众对政府工作的信息透明度并增强政府执政信心等。

(4) 促进服务升级

政府在提供公共服务时,由于缺乏有效的信息预知与实时处理能力和方式,难以实现对实时或即将发生情况的检测和预判,这会导致政府服务的被动性、延迟性、片面性,并且由于缺少对海量个体的单独认知,缺乏个性化的服务模式。实际上,通过大数据背后隐藏规律的提取,政府完全有能力做到全面评估、提前预知、及时防范、有效应对和个性化服务。比如,通过大数据技术对自然灾害(地震、海啸、台风等)的数据进行分析挖掘,可以实现有效的灾害预防、告警并进行妥善的安置管理,降低甚至消除人民群众受灾程度和范围,保护人民生命和财产安全。

(5) 规范权力监督

大数据有助于将权利“关进牢笼”,实现透明、公开的权利监督,主要体现为三个方面:一是可以通过建立电子政务信息化服务平台,将各省、地、市、县、乡各级信息互联互通,

以整合的数据形式推进信息公开透明化；二是可以针对权利执行过程中的行为进行电子化记录，形成有权利、有标准、有行为、有记录、有评估、有奖惩的执行系列联动，真正形成基于数据的政务评估和执政监督体系；三是通过大数据进行权利“全样本”的监督检查和评估反馈，通过统一的数据技术标准对权利执行过程中的各个流程和环节进行统一测量，避免“抽样”导致的权利监管缺失和遗漏。

（6）维护社会稳定

大数据对社会稳定的作用主要体现在预防、分析和应对上。在预防方面，大数据可以基于复杂的社会数据挖掘出每个社会主体的具体关联属性，从源头上做好矛盾化解和安全防卫。在分析方面，基于对所有主体历史和实时数据的分析，可以获得潜藏在社会关系以及个体深层次间的属性关联，由浅入深、由表及里地推导异常越轨行为或关键对象。在应对方面，通过数据提前预测和现场获知的行为事件以及对应的时间、地点、方式、规模等信息有助于政府相关部门及时采取对应的应对手段，以最小化的付出实现社会稳定维持和危险解除。比如，针对恐怖袭击、金融危机等风险对象的历史和实时数据进行分析，找到幕后推动的主要因素，然后针对性地采取措施来监测、控制和防卫其可能带来的危害，增强社会和经济稳定能力。

除了以上方面的应用外，大数据还可以广泛应用于政府选举、科学研究、文化建设等领域，大数据正在支撑履行政府职能、保障公共安全、实施社会治理、支持重大决策、改进公共服务等方面发挥出越来越重要的作用。

11.3 产业价值

在宏观经济增长疲软、增长预期回落，整体经济处于新常态攻坚期的背景下，以大数据信息和相关技术为基础，促进产业结构调整、加快产业动能转换是维持经济健康、可持续发展的关键。大数据对于产业方面的价值主要体现在重塑国际价值链分工、促进产业协同优化、推动产业结构调整、加快产业升级创新四个方面。

1. 重塑国际价值链分工

在国际分工的背景下，中国在改革开放之后依靠廉价的劳动力资源成为国际产业分工中的“世界工厂”，主要优势集中在产品的加工、装配、制造等环节。这些环节在整个国际价值链中的利润率和价值回报较低，关系到产品的研究开发、产品设计、物流运输、批发零售、终端管理等高利润环节则掌握在发达国家手中。从全球价值链看，中国处于价值链的底层，只能获得较少的附加值；而发达国家则处于价值链的高层，获得大部分附加值。大数据对重塑国际价值链分工的作用主要表现在两方面：一是，大数据产业本身属于高端技术产业，其人才、技术、信息、资金等方面的优势使得中国有能力参与到更高端的产业分工和环节中；二是，大数据产业可以助力传统产业的优化、重组、转型和升级，加快其知识引进、技术研发、产品创新、管理国际化的步伐，实现产业自我主导下的跨越式发展。这些都将改变

中国参与国际分工的角色,使中国能够获得更高的附加值,并能提升中国在国际产业链中的地位和影响力,实现由制造大国向制造强国的转变。

2. 推动产业结构调整

产业结构调整是关系到国计民生的重要课题。当前我国以第一产业和第二产业为主,但问题是第一产业农业基础薄弱,第二产业则呈现大而不强的特点,第三产业的核心服务业则发展滞后;再看发达国家,它们已经步入了以信息为基础的知识经济时代,并且依靠知识经济创新了源源不断的国际收益和高价值回报。中国产业结构不合理的问题突出,主要表现在两个层次:第一层次是三大产业之间的结构不合理,第三产业占比很小且产业结构粗犷落后;第二层次是各个产业之间的内部结构不合理,专业化、机械化、智能化程度低。因此,推动产业结构调整极其重要。大数据在产业结构调整中的主要意义表现为三方面:一是通过数据的动态规划来实现资源统筹最大化;二是通过预测需求来反向促进供给计划的配比和生产计划的制定,促进供求协调化;三是通过完整产业链数据的评估,实现对宏观和微观的精准调整,促进经济要素投入产出最大化。

3. 促进产业协同优化

产业链中不同环节分别掌握着各自的信息,同时又对上下游存在信息依赖,将整个产业链数据信息共享和打通才能建立完整的全产业链信息服务。该信息服务可以针对各个环节的各级用户提供信息,辅助于具体运营活动的开展。从横向维度看,产业链环节上的各级用户对数据和信息的需求存在层次性,战略战术层需要整体、宏观的信息,执行层需要局部、微观的信息。从纵向维度看,企业依托产业链上下游信息可以形成反向业务运作能力,将信息转化为自身经营的模式调整、业务优化、个性化运营、产品反向定制等内容;同时,基于完整产业链信息,各链条间的运作协同能形成良性互动,形成价值关联、经营依托、供需协同、空间协作的一体化产业格局。

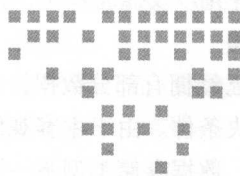
4. 加快产业升级创新

以数据信息引领的知识经济正在深刻地影响着生产要素的网络化、集约化、数字化和智能化,这些将导致新的生产组织方式的诞生,表现在行业中是对传统产业的升级及新产业模式的创新。第一,产业升级。传统产业在大数据的驱动以及产、学、研一体化趋势的不断深化下,正在形成革新的新发展趋势,表现为通过技术再造产业思路、流程、方法和实践。比如,基于大数据相关技术产生的自动驾驶,是在传统人为驾驶以及辅助驾驶基础上的新发展,这一趋势已经开始对汽车产业及交通规划、配套设置等周围产业和政策法规产生深远影响。第二,产业创新。基于大数据激发的商业模式和产业形式的创新,目前主要集中在互联网等新兴领域,这些领域利用丰富和高端的技术流、物质流、资金流和人才流建立的产业模式影响深远。比如,基于个人信用体系建立起的新型消费模式,使得个人信用能够直接与日常生活的各个方向相挂钩,更为重要的是真正将信用和钱关联起来,这在传统产业和商业模式中几乎无法想象。

11.4 本章小结

大数据正在引发一场思维革命，它深刻地改变人们认识世界、观察世界和改造世界的方式和方法，并以前所未有的速度引起政治、经济、社会、文化、学术、科研、国防等领域的变革。整体把握大数据对宏观层面带来的社会价值会让社会各界都认识并参与到大数据事业中来，共同参与才能共同受益。

本章阐述了大数据社会价值的三个方面：民生价值、政务价值、产业价值，需要读者重点掌握的知识点：**大数据对于民生和政务的巨大意义，尤其是民生问题关系到每个人的日常生活，因此需要重点关注。**



大数据当前问题及挑战

越来越多的企业已经认识到大数据的价值，并且考虑着手或已经开始实践利用数据决策和数据驱动来帮助企业更快、更好的发展。但是，企业的大数据实践之路并不是一帆风顺的，各个环节都充满挑战。

大数据工作会存在哪些问题和挑战？本章将从数据、安全、价值、认知、技术和人才几方面进行阐述。

12.1 数据挑战

企业在应用大数据时遇到的首要问题是来自数据本身的挑战。对很多企业来讲，内部没有数据可以使用，或者即使有数据也很难使用的问题非常突出。

缺少数据积累

数据的积累通常是在长期数据规划的前提下进行的，而很多企业（尤其是传统企业）在企业发展之初没有任何数据规划，因此没有数据的积累。企业拥有的数据通常是跟经营核算直接相关的数据，例如财务数据，而客户数据、产品数据、合作伙伴、营销数据等严重匮乏。甚至在某些行业内，很多企业还没有数字化的存储方式，仍然通过纸质等其他媒介进行“存储”。多种多样的存储方式和手段可能也会导致缺乏存储格式、标准的统一管理，这些都直接制约了数据的应用。无数据可用折射出企业在开展大数据时的尴尬。

数据孤岛严重

在很多拥有多业务线的大型企业内，数据常常散落在不同部门。不同部门的数据存储系统、技术、口径、标准、定义等可能都会有所差异，这直接导致了企业数据孤岛的产生。受

制于数据的碎片化以及孤岛效应，企业大数据工作困难重重。

数据质量差

如果企业已经拥有部分数据，也并不意味着能从数据中真正得到价值，数据质量是产生数据价值的先决条件。由于主客观原因，很多数据往往存在各种各样的质量问题，例如数据记录丢失严重、数据存储类型不一致、数据位数不统一、数据字段为空、字段值错误等。即使通过一定手段对数据进行修复，也无法完美再现数据。依赖于较低质量的大数据工作的结论产生和应用都存在巨大的风险。这就所谓的“垃圾进垃圾出”的现象。



如果数据从数据源端就存在质量问题，通常在后期是无法进行有效修复和弥补的，尤其对于丢失、错误、空值类的错误根本无法修复。基于垃圾数据产生的结论可信度很低，某些情况下可能会误导业务方向，这对企业经营来讲将是灾难性的影响。因此，当遇到这类问题时，首先要解决数据质量问题。

数据整合困难

在数据整合应用之前，每个业务体系都有自己的数据规范，这些规范可以有效指导体系内部数据工作的开展。但涉及跨业务体系时，由于在企业全局层面缺乏统一顶层规划而数据整合困难，表现为数据字段定义、关联项、口径、范围、条件、规则等的不一致，再加上数据质量和数据孤岛问题的影响，数据整合和管理应用困难。

数据来源匮乏

当企业内部数据资源较少时，可以考虑从外面获得数据资源。在数据共享趋势下，企业间可以自行或通过特定平台进行数据交换或交易，这是弥补企业数据匮乏的重要方式之一，除此之外还包括爬虫抓取等方式获得数据。但是，大多数可靠的数据源通常基于交换或交易获得，而这些数据范围往往很小，而且大多是已经脱敏或转换后的模糊或粗粒度数据，这对于后续的数据深加工处理和价值提取会造成不利，也让外部数据价值大打折扣。

12.2 安全挑战

大数据安全指的是数据不被损坏、不丢失、不泄露、不被非法查看及非法修改、不给正常运营带来风险等，数据安全意味着数据的可用性、完整性和保密性。大数据安全挑战主要集中在数据物理丢失、数据信息泄露、外部非法入侵等。

数据物理丢失

数据物理丢失指的是数据在采集、存储、新增、删除、修改、更新等过程中，由于主观误操作、客观环境或物理条件等的变化造成的数据物理丢失或损坏。例如：


- 网络延迟、服务器故障、硬盘空间不足、软件编译执行问题、数据采集规划不全面、业务需求变更等导致的数据信息采集的丢失。

- ❑ 存储设备由于感染病毒、设备断电、文件系统错误等逻辑原因，硬盘损坏等物理原因，固件使用次数过多等固件原因导致的数据丢失或缺损。
- ❑ 地震、洪水、雷电、台风等不可抗力因素也可能导致数据丢失，例如东部沿海地区受台风影响造成的洪水泛滥，对服务器存储和应用管理造成一定威胁。
- ❑ DBA 或其他数据访问和管理人员对数据进行删除、清空、批量替换、批量更新等误操作导致的数据丢失及损坏问题，例如 2015 年携程网站及 APP 的瘫痪时间，最大的可能性是内部物理删除。
- ❑ 缺少异地容灾、多重备份、多级容错、磁盘阵列等多级数据存储与防护机制，当系统某部分发生故障或错误时，影响整个系统的正常运行。

数据信息泄露

数据信息泄露指的是企业内部有关核心竞争力的机密资料和重要数据通过一定途径被非授权主体获取。比如，在关键项目投标前已经获得对方标的，在谈判中已经知晓对方底价，在市场发展中提前获得竞争对手的发展方向，在市场并购中提前知晓对方收购对象等都会对企业经营带来巨大负面影响，甚至某些涉及企业核心竞争力以及高度机密的数据泄露会决定企业的兴亡。造成数据信息泄露的主要原因包括：

- ❑ 内部人员的无意泄密。由于系统中毒而导致的信息资料被窃取、无意间将企业内部信息携带到外部环境、与外部人员沟通时无意识的数据披露等都会导致泄密情况的发生。
- ❑ 内部人员的有意泄密。内部人员出于利益、权利和其他关系考虑，有意以公司数据做交换而导致信息泄露。
- ❑ 数据权限失控。企业不同级别人员和角色的权限管理不当，造成访问数据范围、查询权限、导出和保存管理权限的失控等。
- ❑ 加密管理问题。企业内、外部流转的数据文档、文件等，需要根据不同的应用场景和权限做加密处理，如果加密管理制度缺乏良好的实施和管理，也会造成数据泄露。

 **注意** 数据信息泄露的对象不仅包括企业外部主体（例如个人、企业、组织等），还包括企业内部人员。凡是被不具有知晓权的人员获得信息都属于数据信息泄露的范畴。

外部非法入侵

企业间以及部分个人对企业的非法入侵永远是安全和运维管理人员的重要课题，这是一场从未停止并且看不见的战争。他们通过技术手段入侵企业内部的数据节点以获得关键信息，然后非法利用这些信息实现特定目的。

造成外部非法入侵的主要原因通常包括用户口令安全等级低容易遭到暴力破解，数据库、平台、软件、磁盘等功能漏洞，企业防火墙安全等级低，数据存储、传输和分发过程中未加密或加密层次低，SQL 注入等。

除了上述原因外，越来越多的企业开始参与到数据共享与开放的大潮中，客观上也增加了关键数据被非法利用或超限使用的风险。比如，企业的关键用户行为、上下游关键合作伙伴信息、产品信息等在交换过程中如果未经脱敏、转换便直接对外输出，那么很容易造成关键信息的泄露。因此，企业要权衡数据开放和关键隐私保护、数据开放时间和开放程度、满足数据开放需要和保护企业安全等多个方面并找到均衡点。



注意 在大多数场景下，数据安全是企业开展大数据工作的前提条件。换句话说，如果不能保证企业数据和信息安全，那么大数据工作可以停止开展。因此，数据安全具有对大数据工作的“一票否决权”。

12.3 价值挑战

企业对于任何一项经营活动的投入与否，通常基于对未来的预期回报及现实利益的推动的考虑。但大数据工作的价值产出，正遭到来自多方面的挑战。



提示 在所有公司内，几乎都无法直接衡量大数据工作带来的价值，尤其是无法进行财务量化，这是阻碍大数据价值评估的最大挑战。谁都无法回答这样的问题“如果没有大数据，企业会损失多少钱？”大数据工作价值的模糊性，归根到底是其无法独立于业务体而单独存在，只要存在数据依附于业务的现象，数据就永远无法单独衡量。

大数据到底能为哪些公司服务？

大数据的应用可以覆盖所有行业，金融、电信、能源、证券、烟草等大型企业尤为突出。但是，很多企业从数据量的角度来衡量，根本算不上大数据，充其量也就是小数据或中数据，因此大数据对中、小企业的现实价值很小。换句话说，大数据通常只适用于拥有大量数据的企业，而这些企业都是大型或超大型企业、集团。应用主体的规模限制来自于大数据属性本身。

大数据到底能用在哪儿？

大数据似乎可以用在企业日常运营的方方面面，但总有一些领域是大数据规律无法触及的：

- ❑ 大数据通常无法从非常稀疏或罕见事件中得到有价值的规律。比如，在数据预处理过程中，通常会进行异常值处理，罕见事件下对应的异常值通常都是被忽略的；即使是围绕针对异常数据的大数据挖掘，也很难在海量数据中得到几条数据的规律——数据的产生过于偶然，从数据角度看毫无规律可言。
- ❑ 大数据通常很难处理有关情感、关系、上下文情景、背景环境等社会化认知的知识。计算机的特长是针对海量数据的显性规则计算，空间、维度、数量、距离、长度等是其计算的基础。但在面临掺杂了更多的社会化属性的问题时，计算机却难以理解、分

辨、计算和表达这些关系。

- 大数据无法处理很难量化的实体和属性。很多实体和属性无法直接用数据衡量，例如当我们在识别一个人时，可能会说“这个人像张三”，那么这个像的程度到底是多少？这些连人类自己都无法说清楚的东西，大数据更是难以衡量和测算。

大数据到底能带来什么？

在大数据工作流的末端，产出可能是一张报表、一个数字、一幅可视图像或者一段人工智能结论。但究其根本，它们到底是什么？从结果意义上大体可分四类：

- 计划预测。它是对未来的预估和推断，常被应用在业务执行前的计划和评估阶段，预测结果可能是具体数值，或者是一个类别。比如，回归或分类属于这方面的应用，解答“是什么”的问题。
- 结论定义。它是对正在发生的现在和已经发生的过去做出结果判断，以评估结果是否符合预期或存在异常情况。比如，异常检测属于这方面的应用，解答“是什么”和“怎么样”的问题。
- 规则提取。它是从数据中提取频繁规则，以便于指导业务应用实践。比如，频繁项集属于这类的知识，解答“是什么”和“怎么办”的问题。
- 探索性知识。它通常是在没有明确的先导结论和经验下，围绕某一范围或主题开展探索性质的潜在知识挖掘。比如，聚类属于这方面的应用，解答“是什么”的问题。

但上述意义中，少了一类典型且非常重要的意义类型，那就是“为什么”。“为什么”指的是到底是哪些因素导致结论的发生，其中关键因素有哪些，各自的因果联系程度如何等。比如，某公司昨日订单量增加15万单，到底是什么原因导致这一变化：是流量质量好？是促销流程策划合理？还是页面体验效果好？还是促销商品折扣高？大数据通常很难解答这一类的问题，而这一类的问题是几乎所有公司都会遇到的疑问。

提示 可能有的读者会想到，通过多变量测试来控制某些因素而对其他因素进行测试，或者基于非参数检验、降维分析等分析方法可以从中提炼关键影响要素。对于第一种测试方法，由于无法回归已经发生的业务场景且环境体的各个要素都在变化，更无法针对全局样本进行多变量测试，因此测试的方法无法解决该类过去已经发生的问题（对宏观类的主体如宏观经济等，无法通过多变量测试来寻找最优经济政策也属于这类问题）。对于第二类分析方法，要做分析的第一步是进行属性数据量化，而所谓的流程合理性、页面体验友好度等无法直接通过数据采集得到，即使是所谓的“专家”打分法也无法客观解释该问题，因此更不必说找到其中的关键要素。

12.4 认知挑战

站在企业的角度，建立正确的数据认知，树立正确的数据观念尤为重要，因为它会直接影

响大数据工作的方向、方法、进度和预期，并对大数据的最终价值评估和应用产生决定性影响。

大数据的结果并不总是公正客观

大多数企业会认为数据会客观地衡量业务结果，但实际上，价值观的问题从数据策划采集开始已经形成，并贯穿到结果输出的整个过程。在大数据的整个工作流程中，人是一直贯穿着整个过程的主体，在涉及关键数据处理的节点包括异常值处理、数据转换逻辑、数据记录的条件选择、数据模型选择和参数调整、数据可视化表现、数据结果解读等环节都需要人工参与，人工参与必然会受到参与者主观价值观的影响而让数据带有明显的价值倾向或结果导向。例如：

- 在异常值处理过程中，将极小值直接去掉，会使得企业的数字明显上升。
- 在选择样本时，选择数据量大的维度可以刻意展示出某种业务的优秀表现。
- 在可视化时，通过坐标轴的处理可以使得下降趋势非常明显。
- 在分析和解读数据时，通过对比特定维度或时间可以得到相反的结论。

通常，数据工作者应该站在中立的立场来从事数据工作，但数据工作者也是普通的公司职员，其立场取决于所处利益部门和体系，独立于利益部门之外将利于产生客观的数据结果。



注意 客观、公正是数据从业者的职业要求，任何基于数据的工作项目都要求从业者秉着客观公正的态度去对待。

大数据需求往往与业务需求对立

大数据工作往往需要具备一定的条件才能满足数据计算和挖掘的需求，这在很多情况下可能会与业务的需求产生冲突。业务方希望得到快速的、及时的、正确的、全面的结论和反馈，然后进行相应的落地动作；但从数据工作角度来讲，数据的采集、处理、计算、挖掘和可视化都需要一定时间来处理，并且数据量越大、模型越复杂所需时间越多。更重要的是，数据量越大，数据结果的可信度越高且数据规律越明显，数据周期和规模太小可能会产生偶然甚至错误的结论。但是，很多情况下这种冲突无法避免。例如：

- 促销活动只有 7 天，业务部门需要每天针对数据进行调优。
- 临时性突发需求，只能通过临时方法采集部分数据，没有历史可供参考或足够的时间来获得完整数据。



注意 数据需求和业务需求的关系需要平衡。从实质上讲，二者的冲突点其实是在于时间要求的冲突。业务上，预先规划、提前备案以及周全的策划；技术上，增量计算、流处理、实时计算、数据抽样、模型算法的选择和调优等都是提供实时或准实时的重要方式。


大数据不能帮你分析问题

在相关系统和工具提前预设的条件和流程下，计算机会自动呈现出关键结果；当我们把

一些数据分析和挖掘算法模式化后，计算机也会通过自动化的流程产出预期规则。

数据作为一种客观实体，其本身不能帮助我们分析问题，而是提供了可供分析和挖掘的“素材”。唯一能让数据发挥作用的是人，包括分析师、工程师等数据从业者，因此大多数企业的现状不是缺少数据，而是缺少能将数据价值活用并为企业提供辅助决策甚至数据驱动能力的“人才”。

但是，即便有了出色的数据从业者的支持，数据就能真正发挥作用并帮助我们解决所有问题吗？——答案是否定的。在数据落地环境中，即使数据从业者的能力再强、经验再丰富，仍然无法完整重现业务场景，这是数据从业者的短板；相反，这种短板却是业务人员的优势所在，他们的业务经历是数据分析和挖掘的宝贵财富，很多数据分析和挖掘计算的思路都可以从他们身上获得启发、解释和关联信息点。因此，数据和业务两种角色的人员是不可分割的整体。

 **提示** 数据从业人员需要与业务人员紧密结合，从需求发起到应用落地验证和再优化的整个过程，业务人员都必不可少，他们的很多业务经验和常识往往能为数据从业者指明方向并降低数据项目的失败概率。

大数据不能一蹴而就

通常，人们期望能尽快看到劳动成果，大数据工作也不例外。但大数据工作的成果从来都不是一蹴而就的，并不是上了大数据项目企业的运营就会产生立竿见影的效果。原因包括：

- 数据的整体规划、架构、采集、整合、处理、计算和输出需要相应周期来完成。
- 现有数据分散程度、异构复杂度、技术实现难度、业务需求复杂度等因素的影响。
- 数据工作文化的应用也需要时间来完成数据与业务的融合和落地。

但这并不意味着大数据工作的成果都是以“年”为周期的。年度汇报的结果通常是大型的里程碑事件，是可衡量、可应用、可交付的关键产物，而在实现这些关键里程碑之前，通常都需要进行任务分解和阶段性总结，这些都会在项目计划和管理时把控。

除了“小步慢跑”式的阶段性开发和成果交付的方式外，还可以借助外部服务商资源。通过跟有经验、有实力的服务供应商合作，采购成熟的大数据平台直接复用到企业自身，以及半成品大数据平台的基础上进行二次开发和深度定制，也是加快大数据结果产出和落地的重要方式。实际上，这种方式已经成为各企业快速步入大数据工作正轨并取得显著效果的有效途径。

“拍脑袋”还是数据决策

基于数据的决策与基于经验的决策（即“拍脑袋”决策）似乎是相互对立的两种方式，长久以来关于采用哪种决策方式的争论喋喋不休。在大数据时代，二者紧密结合、密不可分。

在数据辅助于决策之前，传统的决策方式都是“拍脑袋”，在这种方式的支撑下很多企业都发展良好，这证明了“拍脑袋”决策的巨大价值；即使在大数据时代，覆盖高科技、金

融、制造、物流、能源等各个领域的所有大型企业，似乎没有一个企业是因为得益于数据决策而从无到有发展起来的，这也体现了一个基本事实——企业没有数据可以发展得很好。

但是，上述认知是基于特定背景下产生的。在企业发展初期，多数都是依赖于自身特有资源而迅速完成原始资本积累，例如销售渠道广、资源供应强、商品品质好、物流速度快、技术先进等，这些作为生产力的关键要素在推动企业积累过程中起到决定性作用。但问题是，当企业已经做大做强之后，企业该何去何从？尤其当竞争对手都在使用数据辅助于决策时，企业自身该如何决策来保证决策的正确性？在信息爆炸的今天，只依靠经营者自身的经验和眼光很难每次都做出科学决策，而且即使顶层决策正确，中层和底层的规划和实施也未必完整有效，因此，数据决策必不可少。

数据决策的价值在于通过可量化的指标、海量的信息归纳、潜在的关联属性、快速的计算能力来提供基于数据的理性结果，而人类的决策价值在于通过感性的认知、不可量化的背景、复杂的社会关系和对经营环境的识别做出判断，将二者结合才能做出更科学化、价值化、利益化的决策。这不仅关系到企业发展方向的战略层面，还关系到企业每个运营环节的战术和实施层面。



提示 数据决策不仅是企业经营者的事情，更是参与企业每个人员的事情，数据决策是推动企业管理规范化、精细化、科学化、个性化的重要保障。

12.5 技术挑战

企业大数据的实施过程中，仍然面临很多技术挑战，涉及数据采集与获取、数据存储与查询、数据处理与计算、数据挖掘与学习、数据理解与应用、数据管理与扩展等方面。

数据采集与获取

数据采集错误和自动修复。大数据的来源复杂且多种多样，不可控的第三方或外部来源中往往包含各种错误信息，以文本文件、电子邮件、互联网信息、遥感器等非标准化结构的数据最为严重。对于错误信息的处理大多通过人工方式实现，而由于错误的多变性、不可预测性、复杂性等问题，通过自动化方式进行错误甄别、提炼、修复等的技术比较少。

数据完整性和可追踪性。不同来源的数据属性间大多存在关联关系，这是数据整合的基础；同一字段在不同周期内的来源也可能出现变化；数据工作流程的后期环节也依附于前期输出，如何保证数据在整个过程中不受事件干扰或意外破坏，并能在产生数据意外时，及时通过数据的血缘地图追踪关键节点，把控异常变化的时间、来源、路径、方式等要素，对于数据质量把控具有重要意义。

数据存储与查询

多类型和结构数据的存储效率和成本。传统的数据存储都是基于结构化的数据库，而大数据平台对于非结构化和半结构化的存储支持丰富了数据的内涵。随着物联网、社交网络、

智能终端等的普及和应用,网络日志、视频、图片、地理位置信息等非结构化数据所占比例越来越大,数据类型也越来越多繁多,如何合理高效地存储这些数据并能降低数据存储成本,同时降低通信成本是巨大挑战。

复杂属性和实体的检索查询。大数据平台中针对非结构化和半结构化数据的查询,大多基于文件名、摘要或文字内容等信息进行检索,对于图片、视频、音频等非文字类的内在语义检索和查询仍然很难实现。除了数据本身的复杂性,多源异构、多实体属性和多维空间之间的交互动态性和关联查询特征,仍然无法通过当前技术进行有效的描述、存储、度量与统一检索。

数据处理与计算

数据计算的实时性。传统的大数据处理侧重于离线计算,但在线计算、实时计算等需求日益显著。由于计算机硬件资源是有限的,即使采用分布式、并行计算框架仍然无法突破硬件的瓶颈,无限增长的数据规模对数据实时处理提出了更高的要求。尤其在面对时间性较强和决策周期较短的场景下,这种需求更加明显,例如地震预测(秒/分钟级)、火灾分析(分钟/小时级)等。

多类型数据的处理与理解。对于隐藏在半结构化和非结构化数据中的潜在信息,通过表征意象提取表意信息,然后将多种来源和类型的数据整合在一起,才能得到全面的数据认知,这是后续进行数据挖掘和深度学习的基础。

数据处理过程中的技术挑战。大数据在处理过程中,面临的噪声处理、样本选择、数据转换、降维归约、算法适配等问题,仍然是通过手工方式实现选择、调整和设置,然后计算机根据人类的设定进行运算。这些关键点在提高人工先验经验参与的同时,也不可避免地带来了人类主观性或经验局限性的影响问题,如何技术化这一过程是解决人工经验或局限性短板的重要课题。

数据挖掘与学习

算法本身的优化。大量的算法由于计算复杂度高导致运行时间较长,无法满足实时性的业务需求;另外,很多现有算法的计算过程,并不适合改造成高度并行化和分布式的运行逻辑。这对算法本身的优化甚至创造提出了更高的要求。

非标准化知识挖掘。现有的机器学习、数据挖掘、深度学习等算法大多是应对标准化的工作任务或场景,因而对数据输入和输出都有对应的训练标准和要求。比如,监督式学习需要提供计算维度和目标标签。但在大量的实际场景中可能面临非标准化或个性化的价值挖掘需求,例如在非监督式的学习中提炼不确定性(没有先验经验模式)的知识和可变的规则等。

高度智能化计算。在知识挖掘过程中,仍然需要人类参与模型配置、训练、评估等过程,计算机本身承担的“仅仅”是在预设条件下进行的大量迭代、更新、推演等。当面临的场景越来越多或越来越复杂时,人类的经验总会存在不确定性、误判性或多变性,这会对计算结果的可信性、准确度和稳定性产生影响。如何让计算机在现有基础上,针对现有数据样本的特征,“自我”调整、配置、计算、评估才是数据智能的核心。

数据理解与应用

丰富的可视化。在数据工作的末端，数据可视化是帮助终端客户理解数据过程和结果的关键手段。传统的可视化主要通过三维以下的图、表等方式展现，并且数据容量较小。在海量数据规模下却很难应对，例如如何展示超过三维维度关系？如何在一个界面中展示超过亿条数据？如何让用户在一个包含 1000 万个样本的复杂图像中直观地看到异常信息？

标准化的数据表达。不同的技术方式输出的数据可能具有各自的定义规则 and 标准，尤其对于不同大型商用工具而言，直接采购的工具往往都缺乏较好的可定制性和二次开发性，由于缺乏统一的标准，当数据在不同工具间流动时会遇到问题。

标准化技术解决方案。当前的大数据技术和产品主要侧重于提供大数据挖掘分析工具和开发环境，而缺少针对各个行业、场景和应用的技术封装，尤其是没有解决不同工具之间的模型和应用流通性，这些问题既有来自于各个国际厂商技术和生态封闭性的原因，也有来自于企业自身在规划大数据架构和开发应用时考虑不周全的原因，这会导致大数据技术解决方案被技术厂商或服务厂商所捆绑，为后期运维、应用、迁移埋下隐患。

数据管理与扩展

随着业务的发展与需求的不断变更，以及技术方案的不不断升级换代，原有的技术方案往往都会面临管理和扩展方面的挑战。一方面，基于数据、硬件、软件、功能权限等方面的多租户管理的需求日渐盛行，客观上要求软硬件和功能开发商灵活性更高、可拆解性和可组合性更强；另一方面对于升级换代过程中面临的功能扩展、技术扩展、接口扩展、服务扩展等方面也提出了更具高瞻远瞩的要求。

12.6 人才挑战

任何大数据工作最终都要靠人来实现，因此熟练掌握大数据工作理论、方法、技术、条件并且具有一定经验的人才正成为各行各业的核心竞争力。但现实情况是，2013 年才是中国大数据的元年，从大数据引进、产生到应用落地的时间也才几年，中国在这方面的积累比较薄弱。

从实践上看，除了 BAT 等几个超大型公司具有产生大数据场景、需求和技术的推动条件外，其他公司都不具备这方面的条件，更不必说基于真正大数据（例如 PB 以上）的工作经验。

从技术上看，大数据的主要技术都来源于国外，中国真正独立开发并能广泛应用的核心技术框架和组件少之又少；另外，技术创新仍然没有成为各个企业创新的焦点，内部创新仍然缺乏广泛的企业土壤。

从人才储备上看，除了个别大型企业集团外，高校和教育机构也开始增加对大数据相关课程及培训体系的投入力度，这对于大数据人才的积累具有重要意义。但是大数据人才从培养到使用需要几年的时间来进行理论积累，更需要深入到大数据工作场景中进行理论验证和

技术落地，因此造成人才供应的不足。

12.7 本章小结

大数据不仅是技术的体现，更是战略、布局、思维、能力和认知的体现。在互联网和大数据推动的多维、跨界、共享、融合、协同创新的潮流下，企业充分认识到大数据当前各方面的挑战，并积极做出调整和自我变革是大势所趋。

本章需要读者重点掌握的知识点包括：

- 安全挑战对于企业大数据的关键意义；
- 大数据工作中涉及的技术挑战。

大数据未来趋势

根据美国著名咨询公司 Gartner 的统计, 在 2015 年, 全球大数据方面的投资依然在增长, 具备为大数据投资的企业比例比 2014 年增长了 2 个百分点达到 75%。由于中国的大数据起步比美国要晚, 很多中国企业还没有开始进入到“大数据时代”, 因此中国的这一趋势会更加明显。

对于大数据的各个应用主体(个人、企业、政府等)来讲, 大数据未来的发展趋势会有哪些? 这些趋势对各个行业、不同政府部门及组织又有哪些影响? 本章将尝试解读这些内容。

13.1 价值资产化

随着拥有数据的规模越来越大且预期价值越来越高, 数据资产化的呼声越发明显。所谓资产, 指的是由企业过去经营交易或各项事项形成的, 由企业拥有或控制的, 预期会给企业带来经济利益的资源。从定义上看, 资产具有三个特征:

- 所有权和控制权归企业所有。
- 基于过去的行为产生的现实资产, 而非预期资产。
- 能够为企业带来经济利益, 包括现金及其等价物。

反观大数据本身, 从现在的实际发生来看, 只有第三条特征尚未完全具备, 这意味着很多企业尚未从大数据中获利, 而且获利是需要能够用货币来计量的。不妨先来看下整个数据生命周期的价值曲线, 如图 13-1 所示。

数据的价值按照生命周期的发展可分为五个阶段: 导入期、形成期、发展期、爆发期和成熟期。

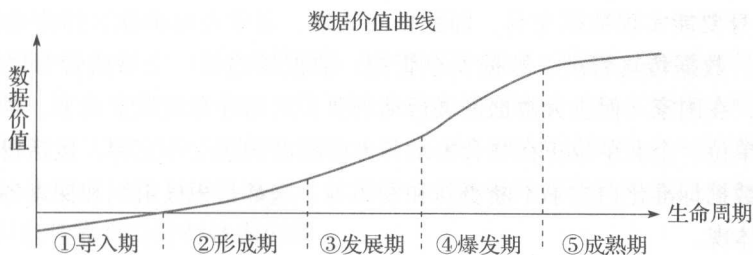


图 13-1 大数据价值曲线

- ❑ 导入期。该时期是数据产生的初期，此时人们主观上并未意识到数据能产生什么作用，或由于客观原因无法从数据中获得价值。因此，数据并未产生任何价值，但却可能需要投入一定的资源进行采集和维护，此时数据是一种“负面资产”。
- ❑ 形成期。由于企业经营环境的改变以及科学技术的发展，在一些行业中率先出现了数据的基础应用，尤其是在电信、金融等数据获取便捷且更具应用需求和场景的传统领域。此时的数据应用侧重点在于报表统计和基础分析，商业价值在于辅助决策。
- ❑ 发展期。在精细化运营、科学化管理以及财务优化的驱动下，对于结构化数据的深入挖掘成为重要方式。此时的典型应用是商业智能类、数据挖掘类、精准营销类等，主要价值体现在辅助商业决策以及浅层次的数据驱动。
- ❑ 爆发期。随着科学技术的进一步推动以及大数据周边产业的不断完善，各行业对全样数据（不抽样）、全类型（结构化、半结构化和非结构化）数据的探索和应用不断增加。此时的典型应用包括个性化推荐、智能管理、语音识别、指纹识别等传统商业领域，以及智能城市管理、智慧交通、智能家居等政务和民生领域，数据的价值已经开始从辅助决策向数据驱动转型，数据的价值也已经可以直接用货币衡量。
- ❑ 成熟期。随着大数据理论、技术、应用等方面的不断进步，未来数据应用会进入到成熟期。在这一时期，数据将作为各数据主体（企业、行业、政府等）常规价值产出的动力之一，并且是保值增值的重要手段。

当前，国内企业的大数据价值阶段仍然分布在除成熟期之外的各个时期，缺少大规模的基于数据资产的商业价值应用。

数据有成为资产的可能性，但不是所有数据都具备资产的属性。要实现数据资产化，需要明确数据资产包含哪些内容。目前的大数据资产，主要指的还是数据本身，但从数据变现的角度，数据资产至少包含数据本身、数据开发和计算技术、数据应用模型和数据衍生服务四类。

大数据本身

大数据本身可以作为商业交易的主体，因此是数据资产最直接的体现。抢占更多的数据资源，拥有更多的数据已经成为数据资产的基石和前提条件，“巧妇难为无米之炊”，缺少数据资源的主体很难获得数据相关的有价值资产。

大数据本身要能实现数据交易，即数据的流通，需要有标准化、体系化的数据质量评估、质量认证、数据传输管理、数据安全管控、数据源追溯、交易监管和保障等方面的支持。幸运的是，在国家《促进大数据发展行动纲要》的指导和政策推动下，各政府部门、学术机关、科研单位、企业单位正在联合推动各方面标准的建立和完善，包括贵阳大数据交易所的建立、大数据标准化白皮书不断更新和发布等，这都是积极组织 and 探索各种大数据交易标准和模式的体现。

大数据技术

大数据技术作为推动大数据产业发展的主要引擎之一，是各企业尤其是高科技企业的核心竞争力。当前大数据技术大多来源于国外，包括 Hadoop、Spark 和 Storm 等，很多大型公司例如 Oracle、Intel、Microsoft、Google、Facebook 等也提供商用版本的大数据技术和产品；而国内的公司只有少数巨头有实力进行独立的技术开发，例如百度、阿里巴巴、腾讯等。大数据技术创新的方式除了独立创新外，还包括联合创新和引入创新。

□ 独立创新：依靠自身力量独立进行技术创新和创造。

□ 联合创新：依靠多企业、多产业、政企结合、产研结合等方式进行联合创新。

□ 引入创新：先引入已有的技术方案，然后基于已有的方案进行创新。

另外，很多公司还投入到开源大数据相关技术的贡献之中，这些技术涵盖大数据工作流的方方面面，对于大数据行业的发展起到巨大的促进作用，尤其是百度、Google、Microsoft、IBM 等开源的有关机器学习、深度学习方面的计算框架和技术，正在推动数据深度价值挖掘的飞速发展。

数据技术创新的前提条件包括硬件、理论和科学技术的进步，例如在分布式大数据系统出现之前，神经网络算法的理论早出现，但受制于没有合适的处理框架实现大规模并行处理、分布式数据存储等问题，导致神经网络的效果无法得到有效验证，最终影响其理论的发展。

对于技术类的数据资产而言，由于技术的周期性和时效性较强，因此需要不断进行技术创新，其推动力大多来源于科学研究推动或业务场景推动。

(1) 基于科学研究的创新

世界上大的科技公司每年都会投入高达数十亿美元进行科学研究，主要用于支持自己的强大研发机构和团队的创新实践，使企业保持旺盛的创新活力以及对新业务、新模式和新市场的拓展。在这一领域内，谷歌（Google X 实验室）对科学研究的创新投入有目共睹，除了围绕其自身服务进行的创新外，谷歌还投入巨资研究无人驾驶汽车、Google Glass 等，当今的谷歌无人驾驶汽车已经获得美国颁发的首例驾驶许可证。



从 20 世纪末开始就已经有汽车集团开始研究 ABS、ACC、车道保持、盲点预警、自动启停、行人检测等辅助驾驶技术，并且在 2010 年奥迪无人驾驶汽车 TTS 就已经行驶 12.42 公里抵达落基山派克峰顶。但这些离真正的无人驾驶还很远（Google 的无人驾驶汽车也没有量产），但无人驾驶的脚本离我们越来越近。

(2) 基于业务推动的创新

很多企业在发展过程中,往往都是被业务“倒逼”着进行创新。淘宝大数据技术的飞速发展便来源于此。读过《淘宝技术这十年》的读者应该都知道,淘宝网从 2003 年 5 月 10 日上线到年底时的每日 3371 万元交易额,在 2015 年双 11 当天已经到达 912 亿元交易额,在巨大业务需求的推动下,基于开源的技术、直接购买最好的商用技术解决方案都无法满足需求,因此淘宝只能依靠自身进行技术创新。

大数据应用模型

大数据应用模型是数据应用和落地的载体,一个应用模型可以是一个算法包、一段脚本、一个工作流甚至一个封装程序。应用模型是基于各细分行业和场景产生的通用方法论和工作标准,它是大数据应用工作的经验总结。

应用模型包含计划管理、效果预测、风险预警、异常管控、规则提取、画像标签等,它为大数据应用消费者、场景实施者提供开箱即用的数据解决方案,能方便、快捷地将数据转变为洞察和价值;更可以基于标准模型的经验进行应用模型的二次开发,结合行业场景、公司背景、业务需求等进行深度定制;另外,应用模型还能促进知识的沉淀和数据应用知识体系的管理,最大化减少员工异动对数据价值输出的影响。

对于大数据应用模型数据资产而言,需要构建企业级大数据中心及知识管理中心,实现数据采集、处理、共享和应用的服务共享,并建立各组件、应用间的松耦合关系,通过数据共享层实现快速数据建模、分析、共享和应用及可视化管理。

如表 13-1 所示为某针对特定行业的部分数据应用模型,每个模型背后对应是数据工作流。

表 13-1 数据应用模型

行业	应用模型
电力行业	防窃电防漏电实时监测,电力系统负荷预测,电力系统故障分析,居民用电分析,电力客户细分,变电设备预警及故障诊断,配电网低电压实时监测,配电设备负载估算及重过载预警,线损计算与精益化降损
农畜牧业及乳业	乳制品配方优化,牧场饲养方案诊断,乳制品市场舆情分析
生产制造	供应链优化,采购价格预测,设备预防性维护预警
炼油炼化	石油炼化裂化工艺优化,设备预防性维护预警
零售业	会员精准营销,会员流失分析,会员忠诚度分析,生命周期分析,销售预测,商品关联分析,会员 RFM 价值分析,会员聚类分群,库存实时调拨推荐
金融保险	反欺诈监测,信用等级评估,客户细分,精准营销,生命周期分析
政府相关	纳税评估,个人信用评估,企业信用评估,公安犯罪侦查
教育科研	职位机会匹配
电信行业	离网客户预测分析,客户价值分析,客户信用度分析,精准营销,客户细分
电子商务	购物篮分析,网络入侵监测,网络广告定向投放,个性化推荐,客户细分,精准营销,生命周期分析

大数据衍生服务

大数据衍生服务包括由内向外传递的衍生服务、基于技术的外部衍生服务、基于咨询的外部衍生服务、基于周边支撑的衍生服务四类。

(1) 由内向外传递的衍生服务

由内向外传递的衍生服务指的是数据工作主体在自身能力、规模、经验和技術达到一定阶段之后，基于自有数据或技术资源，直接将现有成型的内部数据（包括原始数据、脱敏数据、数据结果标签）或解决方案提供给外部使用，即为第三方提供服务。典型如阿里巴巴，在自身拥有大量数据和技術的前提下，将部分内部技术进行整合并输出为阿里云、店铺管理、广告竞价等各项数据服务模式，涵盖了数据管理监控、数据统计分析、云存储、弹性计算、大规模计算等各种 SaaS 服务。

(2) 基于技术的外部衍生服务

很多大数据公司自身并没有数据，但其依靠自身成熟产品、技术实力和行业经验，为客户（数据拥有者）提供大数据相关服务，包括数据采集服务、数据挖掘和分析服务、数据应用服务、数据交换服务等。比如，百分点科技本身没有任何数据，但通过为客户提供推荐服务获得大量用户行为数据，基于这些行为数据再与第三方公司进行数据整合，然后对所有用户行为进行用户画像分析，得到进一步有价值的画像系统、用户分析系统等，提供分析、咨询、推荐、营销等方面的服务。

(3) 基于咨询的外部衍生服务

很多大数据公司对外提供的是大数据服务或解决方案，涉及整体规划、组织架构、应用咨询、人员培训、数据建模、数据分析、商业模式转型等。随着大数据需求越来越多但专业人才缺口加大，也受制于特定单位的人员编制限制，通过外部公司提供大数据服务模式非常流行。这类服务的形式通常是项目外包、人才外包或项目咨询。比如，麦肯锡是一家管理咨询公司，它的客户均为各领域优秀的大型公司，涵盖能源、汽车、银行、保险、制造、公共事业、零售、电信、交通等领域，他们提供的咨询用来指导客户的经营实践，并对最终成果质量负责。

(4) 基于周边支撑的衍生服务

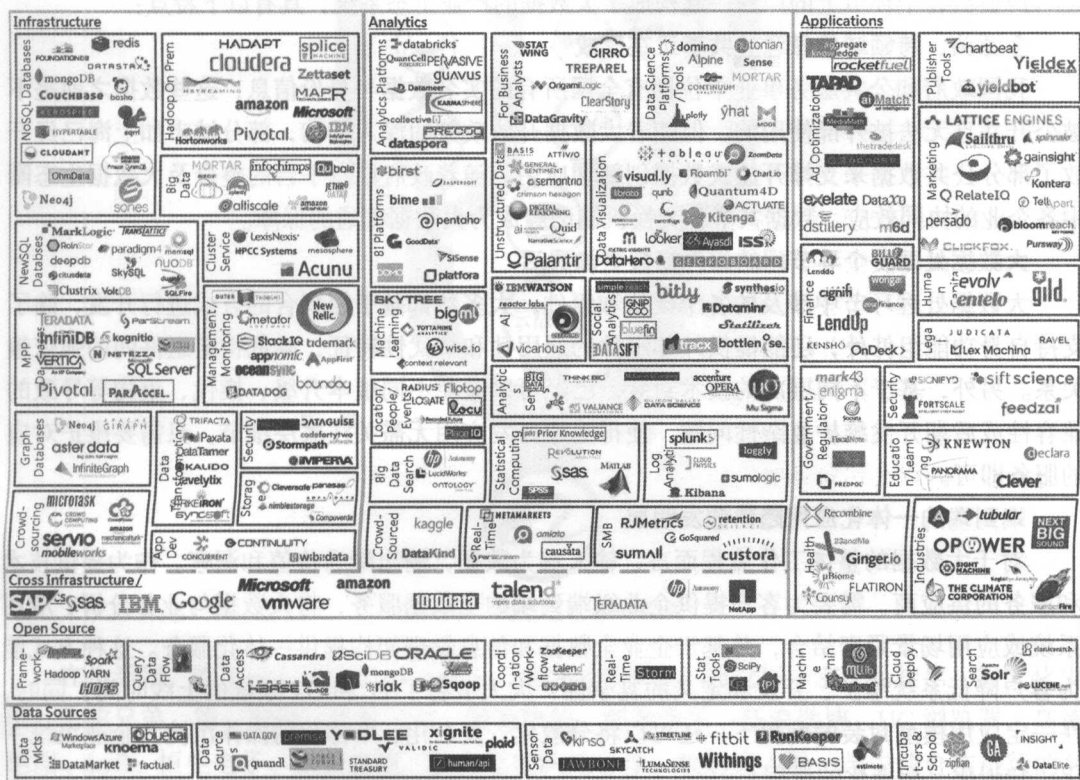
大数据衍生服务除了本身 workflow 涉及的各环节外，还包括周边配套服务。

- ❑ 市场化交易集市：在由内向外传递的衍生服务中的数据交易是数据个体间的行为，但市场化数据集市则是将所有可控交易的数据放到一起，形成统一的数据采购和交易平台。
- ❑ 培训认证：可以为对外提供数据服务的个人或企业提供培训和资格认证，包括技术培训和培训。
- ❑ 征信评价：在大数据交易过程中，对于交易多方主体的信用等级的评估。建立基于数据征信评价的个人、企业的征信系统，并与国家征信系统结合。
- ❑ 其他：在整个大数据市场交易过程中的监督、管理、标准制定等服务内容。

13.2 产业生态化

在当今的大数据产业中，各个参与主体承担着不同的角色和职能。大数据的产业链贯穿数据的工作生命周期，即从数据的产生、采集、传输、存储、分析、挖掘，直至最终的可视化呈现与行业应用。如图 13-2 所示，大数据产业链中包含了基础设施、分析、应用、开源工具、数据源和 API、学校和孵化基地等。

BIG DATA LANDSCAPE, VERSION 3.0



© Matt Turck (@mattturck), Sutan Dong (@sutandong) & FirstMark Capital (@firstmarkcap)

图 13-2 大数据产业链景观图

- **基础设施:** Hadoop、Spark、集群服务、NoSQL 数据库、NewSQL 数据库、图数据库、MPP 数据库、云企业数据仓库、数据转换、数据集成、管理监控、安全、存储、APP 开发、众包等。
- **分析:** 分析师平台、分析平台、数据科学家平台、数据可视化、BI 平台、统计计算、日志分析、社交分析、实时分析、机器学习、语音和自然语言处理、横向 AI、检索、数据服务、商业分析、SMB 电子商务等。
- **跨基础设施和分析:** 提供综合服务。

- 应用：销售和市场、客户服务、人力资源、法律、广告优化、社会、垂直 AI 应用、出版工具、政府监管、财务、教育学习、生活艺术、行业等。
- 开源工具：基础框架、请求和数据流、数据接入、调度、实时处理、统计工具、机器学习、检索、安全、可视化等。
- 数据源和 API：健康数据、物联网数据、财务和电商数据、航空/空间/航海数据、位置/人群/实体数据、其他数据等。
- 学校和孵化基地。

产业链各个环节上的厂商一起构成了大数据的产业生态系统，具有以下特点：

多源头、跨领域信息源的共享正在形成

我国政府和公共服务事业单位拥有全国所有的实名数据与身份信息，这些数据之前是不被公开并且无法被外部使用的。但随着大数据开放政策的深入贯彻，部分城市如上海已经开放了部分公共数据来支撑开放式数据创新和应用。随着政府信息平台的整合及大数据生态圈中各企业的协同效应，数据共享和交换机制已经成为大数据流通的保障。

大数据处理各个环节间的组件衔接更加灵活

大数据处理环节中涉及各个模块和组件，大多数都基于松耦合的方式进行开发，除了支持自身功能组件外，还能提供外部组件的通用性和替代性特征，降低特定组件间的强依赖关系。另外，基于不同层次的技术封装，已经可以避免不同版本升级、换代、重构等导致的兼容性或后期开发维护复杂性问题，使得上层开发应用无需关注底层细节，只需要维护对应的服务即可。

端到端的一体化应用趋势愈发明显

对于大数据的整体工作流程而言，输入端为数据，输出端为价值和洞察。作为提供技术和服务的供应商，需要为客户提供企业级端到端的全数据服务，将大数据与业务分析、应用系统或应用场景紧密结合，并基于企业全数据和信息管理架构提供一体化服务。这种方式能为客户降低系统维护、开发和定义的复杂性，大大节省了 IT 和 DT 系统的上线时间；同时，在一定应用模型封装的前提下，还可以将分散化、复杂化、个体化的操作沉淀为企业级统一化、知识化、标准化的流程。

跨界数据服务的流通和应用逐步加剧

大数据企业在扩大自身数据规模、实现内部数据流通的基础上，还可以通过跨界合作打通数据和相关业务链条，实现数据资产战略驱动主营业务的增长。以图 13-3 阿里巴巴投资版图为例，阿里巴巴正在将自己的传统影响力从消费端引导到生产端，形成生产→运输→消费的产业链条。从现有格局看：

在消费端，通过天猫、淘宝、聚划算、一淘等控制线上个人消费；通过阿里巴巴获得企业间的消费和交易市场；同时，通过与银泰、美的、海尔等的合作渗透到线下消费。

在金融端，从个人消费的支付宝，渗透到中小企业的网上银行和蚂蚁金服。

在物流端，组建菜鸟网络，对接美的、海尔等传统品牌商以及“四通一达”的物流体系，

另外还有中国物流骨干网、新加坡邮政、卡行天下等布局。

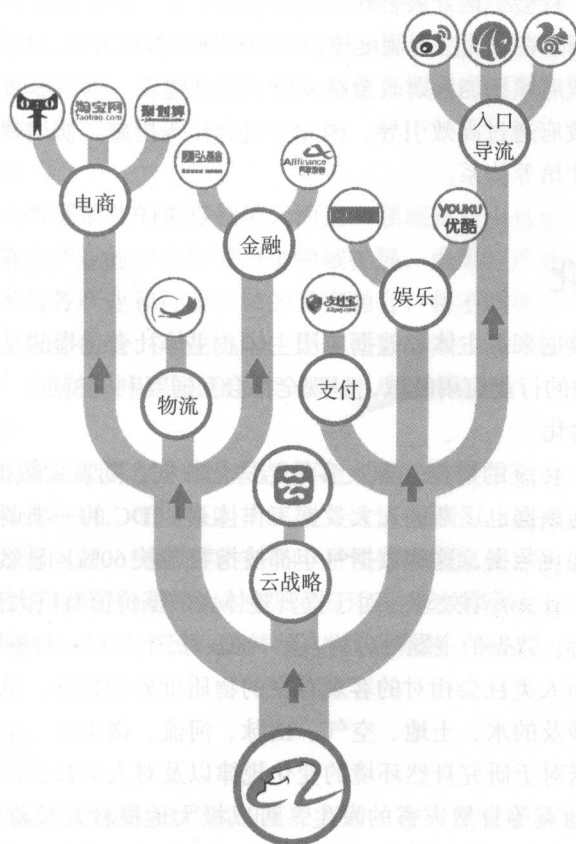


图 13-3 阿里巴巴投资版图

在生活端，除了进军智能生活领域外，组建阿里影业，通过娱乐宝产品推出，对文化中国、华谊兄弟、第一财经、优酷土豆、滴滴快的、虾米音乐、新浪微博、UC 浏览器、高德地图、Lyft、恒生电子、众安在线、恒大足球队的合作、收购或入股，与中信合作进行的药品识别码系统进入到个人健康、娱乐、教育、文化、出行等生活的各个领域。不仅如此，阿里巴巴还通过与政府部门的合作，渗透到公共生活、城市服务等领域。

至此，阿里巴巴已经基本完善了围绕消费、金融和物流的个人消费体系，并逐步扩展到个人生产、生活、学习和工作的各个场景。

产业周围政策和配套服务正在完善

大数据产业和生态圈的完善离不开国家政策的引导和配套服务的完善，包括：

- 数据标准上，国家已经联合相关产、学、研、用等单位制定了《中国大数据交易白皮书》、《大数据标准化白皮书》等标准化工作文件，对数据标准的制定和普及起到示范作用。

- 政策倾斜上，国务院印发的《促进大数据发展行动纲要》指出健全市场发展机制，鼓励政府与企业、社会机构开展合作。
- 法律法规上，国家在加快法规制度建设，积极研究数据开放、保护等方面的规章制度。
- 财政支持上，政府预期加大财政金融支持，推动建设一批国际领先的重大示范工程。
- 人才观点上，政府通过高效引导、国际交流合作等措施，协助建立健全多层次、多类型的大数据人才培养体系。

13.3 主体社会化

大数据主体包括数据来源主体和数据应用主体。主体社会化指的是数据来源和应用主体特征都突破各自单独的行业应用领域，呈现全社会互通互利的特征。

数据来源主体社会化

从数据类型来看，传统的数据来源大多是结构化数据；随着大数据的技术进步与成熟，半结构化和非结构化的数据也逐渐进入大数据工作体系。IDC 的一项调查报告指出：企业中 80% 的数据都是非结构化数据，这些数据每年都按指数增长 60%。显然，加强对半结构化和非结构化数据的来源采集及后期处理，对于提升整体大数据价值有巨大促进意义。

从数据的主题来看，数据的主题可分为自然数据、民生数据、政务数据、产业数据。

自然数据指的是与人类社会相对的客观存在的物质世界的的数据，也可以说是自然界的数据，包括人类居住所涉及的水、土地、空气、山脉、河流、微生物、植物、动物、气象、地球、宇宙等。这些数据对于研究自然环境的变化规律以及对人类社会的影响有极其重要的作用。比如，通过研究地震等自然灾害的发生，可以极大地挽救甚至避免人类的生命和财产损失。

民生数据指的是人民日常生活中的生活行为，包括衣、食、住、行、学习、工作、娱乐、家庭、社会等方面。在传统的大数据来源中，涉及更多的是侧重与经济行为相关的数据，其他的数据来源和利用较少。民生数据是建立完整个人画像的基础，更是国家宏观社会管理的重要参考。

政务数据指的是国家相关机关和单位的立法、司法和行政工作的数据，包括审判、监察、立法和日常组织、领导、计划、人事、协调、监督、财务等。国家正在加强政务部门的大数据建设，并促进政府数据开放，这对于进一步提升政务的智能管理，提高社会化服务水平，加强科学调整有重要促进作用。

产业数据指的是社会经济活动的所有数据，包括原料采购、产品设计、产品加工、仓储管理、物流运输、产品分销、营销推广、终端零售、售后服务等。各产业的经济活动一直是最具活动的领域，这部分的大数据工作也一直在不断创新。

数据应用主体社会化

数据应用即数据消费，指的是使用数据。数据消费的主体包括个人、企业和政府。

(1) 个人数据消费

大数据与个人生活息息相关,越来越多的生活场景中已经渗透了大数据的智慧和结晶。我们生活中涉及的衣、食、住、行等各个方面都可以基于大数据的知识进行决策,个人在这个过程中既是大数据的贡献者,又是大数据的受益者。大数据将为个人提供认识世界、改造世界甚至预测未来的新方式。

(2) 企业数据消费

企业数据消费是大数据消费的重点领域,而这一领域正在由传统的大数据阵地,例如金融、互联网、电信等第三产业的部分领域向全领域扩展,由第三产业向第一产业和第二产业扩展。大数据信息技术与各产业和行业领域的紧密融合,将不断提升企业对大数据的认知和理解,通过技术创新、应用创新、产业协同、集体智慧等方式共同加快各领域的大数据开发与利用,推动企业对于数据的应用需求和发展水平进入新的阶段。

(3) 政府数据消费

国家所拥有的数据规模是其综合国力的重要组成部分,大数据在国防、反恐、安全、国际关系处理等领域的有效应用,可以帮助其解决关键情报缺失、监视范围不完整、侦查能力不足、分析能力不全面等问题,提高国家安全防卫和处理国家关系的能力;同时,在国家内部管理上,大数据在政府和公共服务领域的深度应用可有效推动政务工作的开展,提高政府部门的决策水平、服务效率和社会管理水平,促进社会和谐、科学、持续发展。

13.4 应用智能化

应用智能化是大数据工作和价值产出的重要方向。应用智能化分为三个阶段:

初级阶段:通过设定阈值进行数据智能化工作,阈值的设定支持任意维度、任意指标、任意时间范围,并通过可接触方式进行业务落地,例如智能短信、自动 EDM、桌面提醒等;为了实现数据的直接驱动价值,还可以将数据智能条件控制与业务系统集成,直接完成数据端到用户端的智能应用。

中级阶段:现阶段的数据工作方式是人工确定业务问题并人工开展数据研究工作,所以中级阶段需要能够在人工指定业务问题的基础上让数据自动开展研究工作。当前的机器学习、深度学习、文本挖掘、自然语言处理等都是这一阶段的实现要素,但现有的这些要素都仍处在发展期,还需要人工进行训练、调优或指定规则才能帮助机器实现价值掘金。

高级阶段:当数据能够自动识别业务问题并进行数据研究时,数据就已经具备了类似于人类的“意识”。机器从事数据工作的最大弊端在于缺少人类社会背景信息以及所谓的“社会意识”,如果可以把“社会意识”用数据的方式记录下来,那么通过机器的自我学习和进步,一定会形成机器的“自我意识”和“自我社会观”。此时,机器便可以自动识别并处理任何问题,人类的角色则侧重于机器管理和社会管理。

大数据应用智能化的最终阶段需要具备以下特征:

□ 感知能力：具有能够感知外部环境并获取相应信息的能力，这是产生智能应用的前提。

□ 思维能力：能够对外部获取到的数据进行分析、计算、比较、判断、联想和决策。

□ 进化能力：通过与环境的相互作用，不断积累新知识，使自己能够适应环境变化。

□ 决策能力：通过对外界信息的感知和判断，做出相应的反应并形成决策传达出去。

大数据作为一种从数据中创造和提炼价值的工具，将会在许多领域得到应用，带来广泛的社会价值。虽然，目前多数大数据企业的应用层次比较低，大多处于初级和中级阶段，但最终都会借助于成熟的经验、技术和方法，快速复制和借鉴成功模式来迅速提升自己。

13.5 本章小结

大数据作为企业发展的源动力之一，在推动行业发展的同时也为政府服务和民生服务提供了样板。未来，企业仍将扮演推动大数据价值资产化、生态化、社会化和智能化的主要动力。待数据真正实现资产化之后，数据资产也会相应地进入资产负债表，成为财务三大报表以外的第四极。

本章需要读者重点掌握的知识点包括：

□ 了解大数据未来发展的趋势，尤其是价值资产化和应用智能化；

□ 重点掌握数据资产的内容及落地方向。

作者简介

吕兆星 (EthanLv) 资深大数据技术专家，精通基于大数据的分布式数据挖掘、存储与计算技术，及其生态体系架构；精通垂直搜索技术、机器学习、文本情感倾向性挖掘、网络爬虫、全文索引体系架构。曾任软通动力集团大数据研究院总架构师，HiveCloud创始人，萝卜网CTO，国美在线大数据中心高级架构师等。

郑传峰 (PeterZheng) 大数据业务应用领域专家，主导大数据方向战略规划，包含数据产品、数据应用、数据价值变现等方向。曾任软通动力数据科技公司资深数据应用专家，HiveCloud首席战略官。阶段性负责国美电器、国美在线、库巴网的会员营销、网站运营和产品设计工作，在CRM系统、DMP数据平台、精准营销系统、广告精投、能源大数据等领域拥有多年的操盘经验。

宋天龙 (TonySong) 大数据领域的资深数据分析、挖掘和建模专家，精通端到端数据价值场景设计、业务需求转换、数据结构梳理、数据建模与学习，以及数据工程交付。曾任软通动力集团大数据研究院数据总监，Webtrekk（德国最大的网站数据分析服务提供商）中国区技术和咨询负责人，国美大数据中心经理。

杨晓鹏 (KelvinYang) 大数据及BI技术领域资深架构师，精通传统数据模式及大数据分布模式的数据存储、计算与应用架构，以及大数据量的数据迁移、存储、索引、计算、分析与挖掘等相关环节的设计、实现与优化。曾任软通动力集团大数据研究院高级架构师、HiveCloud总架构师，主导大数据存储平台、计算平台和应用服务平台的设计与研发。曾任居然之家O2O大数据平台总负责人、中国银联大数据报文分析项目高级技术顾问、国美在线大数据中心高级技术工程师。

刘天文 软通动力集团董事长

大数据是推动世界经济发展、加快社会进步、提高民生福祉、促进国家稳定、提高国防安全的重要武器，大数据战略势在必行。本书站在企业级角度，从战略、战术和方法等不同层次阐述大数据的经纬脉络，既有方法论又有落地方案，不限于技术技巧，更提供应用案例和风险前瞻，整体结构分明，层次性强。

杨学成 北京邮电大学经济管理学院教授/副院长

随着互联网的进化，业务数据化的进程明显加速，企业的数据资产管理成为未来商业成功的关键。如何做好企业的大数据规划，如何深入挖掘企业大数据的价值，如何利用数据驱动商业模式蝶变，都是企业经营必须深入思考的战略性问题。本书为上述问题提供了直接、简洁、可操作性十足的解决方案，有助于数据型企业的价值发现，是寻求转型的传统企业的必备参考书！

姜继浩 中国社科院博士/正大集团副总裁

大数据强大的决策力、洞察力与变革力，正广泛应用于BI、工业4.0、云计算、物联网及互联网+等领域。本书有关大数据宏观蓝图的设计和规划，以及微观内核的洞悉和观察，在业界为数不多。这是一本产研结合、讲解透彻、深入浅出、思维全面的书籍，相信它一定会给您带来启发！

傅志华 360大数据中心副总经理

在多年大数据实战的基础上，作者系统总结了大数据的企业应用方法论，非常值得大数据从业者参考。大数据在企业的应用，需要从大数据的战略定位、组织架构、应用设计、技术架构等多个方面做出系统规划，缺一不可。

叶石砚 国美管家总经理/国美在线副总裁

在人们还沉浸于讨论多大规模的数据才是“大数据”，为什么“大数据”可以迅速弥漫于我们的思维、商业和管理领域的时候，本书已经以一种极致的思维分析模式及实用的统计研究角度，对企业大数据模式从0到1再到10的过程进行了全面分析，内容详尽不冗杂，应用更具实操性。

田学峰 八戒教育总经理，萝卜网创始人

大数据不仅是技术，更是战略、战术和思维方法，事实已经证明，大数据在推动民生、社会、经济、政治等各方面的变革。但是到底如何促进大数据落地实施仍困扰着很多企业，本书提供了解决方案，在大数据的广度、深度和高度上都很有见地。

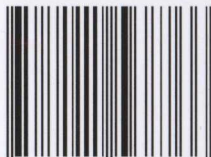


投稿热线: (010) 88379604
客服热线: (010) 88379426 88361066
购书热线: (010) 68326294 88379649 68995259

华章网站: www.hzbook.com
网上购书: www.china-pub.com
数字阅读: www.hzmedia.com.cn

上架指导: 计算机/大数据

ISBN 978-7-111-56876-6



9 787111 568766 >

定价: 89.00元